Non-idempotent intersection types and strong normalisation

Alexis Bernadet, Stéphane Graham-Lengrand

Non-idempotent intersection types and strong normalisation

Alexis Bernadet[1] Stéphane Graham-Lengrand[1,2]
[1]École Polytechnique, France
[2]CNRS, France
{bernadet,lengrand}@lix.polytechnique.fr

5th October 2013

Abstract

We present a typing system with *non-idempotent intersection* types, typing a term syntax covering three different calculi: the pure λ-calculus, the calculus with explicit substitutions λS, and the calculus with explicit substitutions, contractions and weakenings λlxr.

In each of the three calculi, a term is typable if and only if it is strongly normalising, as it is the case in (many) systems with idempotent intersections.

Non-idempotency brings extra information into typing trees, such as simple bounds on the longest reduction sequence reducing a term to its normal form. Strong normalisation follows, without requiring reducibility techniques.

Using this, we revisit models of the λ-calculus based on *filters of intersection types*, and extend them to λS and λlxr. Non-idempotency simplifies a methodology, based on such filter models, that produces modular proofs of strong normalisation for well-known typing systems (e.g. System F). We also present a filter model by means of orthogonality techniques, i.e. as an instance of an abstract notion of *orthogonality model* formalised in this paper and inspired by classical realisability. Compared to other instances based on terms (one of which rephrases a now standard proof of strong normalisation for the λ-calculus), the instance based on filters is shown to be better at proving strong normalisation results for λS and λlxr.

Finally, the bounds on the longest reduction sequence, read off our typing trees, are refined into an *exact measure*, read off a specific typing tree (called *principal*); in each of the three calculi, a specific reduction sequence of such length is identified. In the case of the λ-calculus, this complexity result is, for longest reduction sequences, the counterpart of de Carvalho's result for linear head-reduction sequences.

Contents

1 Introduction

Intersection types were introduced in [CD78, CDS79], extending the simply-typed λ-calculus with a notion of finite polymorphism. This is achieved by a new construct $A \cap B$ in the syntax of types and new typing rules such as the one on the right, where $M : A$ denotes that a term M is of type A.

$$\frac{M : A \quad M : B}{M : A \cap B}$$

One of the motivations was to characterise strongly normalising (SN) λ-terms, namely the property that a λ-term can be typed if and only if it is strongly normalising. Variants of systems using intersection types have been studied to characterise other evaluation properties of λ-terms and served as the basis of corresponding semantics [BCDC83, Lei86, Kri90, vB95, Ghi96, AC98, Gal98, DCGL04, DCHM05, ABDC06, CS07].

This paper develops [BL11a, BL11b] with detailed proofs and extends the results to other calculi than the pure λ-calculus, namely the calculus with explicit substitutions λS (a minor variant of the calculi λ_s of [Kes07] and λ_{es} of [Ren11]), and the calculus with explicit substitutions, explicit contractions and explicit weakenings λlxr [KL05, KL07]:

It presents a typing system (for a syntax that covers those of λ, λS and λlxr) that uses *non-idempotent* intersection types (as pioneered by [KW99, NM04]).

Intersections were originally introduced as idempotent, with the equation $A \cap A = A$ either as an explicit quotient or as a consequence of the system. This corresponds to the understanding of the judgement $M : A \cap B$ as follows: M can be used as data of type A or as data of type B. But

the meaning of $M : A \cap B$ can be strengthened in that M **will** be used **once** as data of type A and **once** as data of type B. With this understanding, $A \cap A \neq A$, and dropping idempotency of intersections is thus a natural way to study control of resources and complexity.

Using this typing system, the contributions of this paper are threefold:

Measuring worst-case complexity.

In each of the three calculi, we refine with quantitative information the property that typability characterises strong normalisation. Since strong normalisation ensures that all reduction sequences are finite, we are naturally interested in identifying the length of the longest reduction sequence. This can be done as our typing system is very sensitive to the usage of resources when terms are reduced (by any of the three calculi).

Our system actually results from a long line of research inspired by Linear Logic [Gir87]. The usual logical connectives of, say, classical and intuitionist logic, are decomposed therein into finer-grained connectives, separating a *linear* part from a part that controls how and when the structural rules of *contraction* and *weakening* are used in proofs. This can be seen as resource management when hypotheses, or more generally logical formulae, are considered as resource. The Curry-Howard correspondence, which originated in the context of intuitionist logic [How80], can be adapted to Linear Logic [Abr93, BBdH93], whose resource-awareness translates to a control of resources in the execution of programs (in the usual computational sense). From this has emerged a theory of resource λ-calculus with semantical support (such as the differential λ-calculus) [BL96, BCL99, ER03, BET10, BEM10]. In this line of research, de Carvalho [dC05, dC09] obtained interesting measures of reduction lengths in the λ-calculus by means of *non-idempotent* intersection types: he showed a correspondence between the size of a typing derivation tree and the number of steps taken by a Krivine machine to reduce the typed λ-term, which relates to the length of linear head-reductions. But if we remain in the realm of intersection systems that characterise strong normalisation, then the more interesting measure is the length of the longest reduction sequence.

In this paper we get a result similar to de Carvalho's, but with the measure corresponding to strong normalisation: the length of the longest β-reduction sequence starting from any strongly normalising λ-term can be read off its typing tree in our system.[1]

Moreover, the idea of controlling resource usage by intersection types naturally leads to the investigation of calculi that handle resources more explicitly than the pure λ-calculus. While the resource calculi along the lines of [BEM10] are well-suited to de Carvalho's study of head reductions, our interest in longest reduction sequences (no matter where the redexes are) lead us to explicit substitution calculi along the lines of [KL05, KL07, Kes07, Ren11]. Hence the extension of our complexity results (already presented in [BL11a] for λ) to λS and λlxr.[2]

Filter models and strong normalisation.

Intersection types were also used to build filter models of λ-calculus as early as [BCDC83].[3] In particular, [CS07] shows how filters of intersection types can be used to produce models of various type theories; this in turn provides a modular proof that the λ-terms that are typable in some (dependent) type theory (the *source system*) are typable in a unique strongly normalising system of intersection types (the *target system*), and are therefore strongly normalising.

Following [BL11b], we show here an improvement on this methodology, changing the target system of idempotent intersection types to our system of *non-idempotent* intersection types.[4] The benefit of that move is that the strong normalisation of this new target system follows from the fact that typing trees get strictly smaller with every β-reduction. This is significantly simpler than the strong normalisation of the simply-typed λ-calculus and, even more so, of its extension with idempotent intersection types (for which [CS07] involves reducibility techniques [Gir72, Tai75]).

[1] While Linear Logic also evolved typing systems that capture poly time functions [Bai02, BM03, Laf04, GR07], let us emphasise that no linearity constraint is here imposed and all strongly normalising λ-terms can be typed (including non-linear ones). In this we also depart from the complexity results specific to the simply-typed λ-terms [Sch82, Bec01].

[2] In particular, the explicit contractions of λlxr relate to the left-introduction of intersections.

[3] For instance, [ABDC06] reveals how the notion of intersection type filter can be tuned so that the corresponding filter models identify those λ-terms that are convertible by various restrictions of β- and η-conversion.

[4] This also departs from [ABDC06].

Strangely enough there is no price to pay for this simplification, as the construction and correctness of the filter models with respect to a source system is not made harder by non-idempotency.

While this improvement concerns any of the source systems treated in [CS07], we choose to illustrate the methodology with a concrete source system that includes the impredicative features of System F [Gir72], as suggested in the conclusion of [CS07].

Moreover, extending our improved methodology [BL11b] to the explicit substitution calculi λS and λlxr is a new contribution that addresses problems that are reputedly difficult: as illustrated by Melliès [Mel95], strong normalisation can be hard to satisfy by an explicit substitution calculus. When it is satisfied, proof techniques often reduce the problem to the strong normalisation of pure λ-terms via a property known as *Preservation of Strong Normalisation* (PSN) [BBLRD96], while direct proofs (e.g. by reducibility [Gir72, Tai75]) become hugely intricated [DL03, LLD$^+$04] even in the simplest explicit substitution calculus λx [BR95]. Here we have direct proofs of strong normalisation for λS and λlxr, when it is typed with simple types, idempotent intersection types, System F types. These are, to our knowledge, the first direct proofs for those systems (i.e. proofs that do not rely on the strong normalisation of pure λ-terms).

Orthogonality models.

The third contribution of this paper is to show how the above methodology can be formalised in the framework of *orthogonality*. Orthogonality underlies Linear Logic and its models [Gir87] as well as classical realisability [DK00, Kri01, MM09], and is used to prove properties of proofs or programs [Par97, MV05, LM08].

We formalise here a parametric model construction by introducing an abstract notion of *orthogonality model*, which we illustrate with three different instances:

- one instance is a model made of strongly normalising terms (which, in the case of the pure λ-calculus, captures the traditional use of orthogonality to prove strong normalisation [Par97, LM08])
- one instance is a model made of terms that are typable with intersection types
- one instance is a model made of filters of intersection types

To our knowledge, this is the first time that some filter models are shown to be captured by orthogonality techniques. Also, the systematic and modular approach offered by the abstract notion of orthogonality model facilitates the comparison of different proof techniques: As already showed in [BL11b], all three orthogonality models provide proofs of strong normalisation for the pure λ-calculus. But here we also show that, in the case of λS and λlxr, the term models fail to easily provide such direct proofs: one has to either infer that a term is strongly normalising from some normalisation (resp. typing) properties of its projection as a pure λ-term (as in the PSN property), or prove complex normalisation (resp. typing) properties within λS and λlxr themselves (as in the **IE** property identified in [Kes09]). On the contrary, the filter model provides strong normalisation results for λS and λlxr as smoothly as for the pure λ-calculus.

Structure of the paper.

This paper aims at factorising as much material as possible between the three calculi, and present the material specific to each of them in a systematic way.

Section 2 presents the generic syntax that covers those of λ-calculus, λS and λlxr; it presents the (non-idempotent) intersection types, the typing system used in the rest of this paper and its basic properties.

Section 3 proves Subject Reduction for each of the three calculi, showing that typing trees get smaller with every reduction, from which strong normalisation is inferred (*Soundness*).

Section 4 presents the filter structure of our intersection types, and the construction of a filter model for a very general *source typing system*, which is thus proved strongly normalising in each of the three calculi; the abstract notion of *orthogonality model* is defined with sufficient conditions for the Adequacy Lemma to hold (being typed implies having a semantics in the model); three instances of orthogonality models are defined and compared in the view of proving strong normalisation results for the three calculi.

Section 5 proves Subject Expansion for each of the three calculi, from which typing derivations are shown to exist for every strongly normalising terms (*Completeness*); such derivations are proved

to satisfy some specific properties called *optimality* and *principality*.

Section 6 draws advantage of the optimality and principality properties to refine the upper bound on longest reduction sequences into an exact measure that is reached by some specific reduction sequence; this is done in each of the three calculi.

Section 7 discusses alternative measures for the explicit substitution calculi λS and λlxr, and Section 8 concludes. An appendix details the proofs of the theorems that would otherwise overload the paper with technicalities.

2 The calculus

The intersection type system we define here was first designed for the pure λ-calculus. However, it can easily be extended to other calculi such as the explicit substitution calculus λS, or the explicit substitution calculus λlxr where weakenings and contractions are also explicit.

The theories of those three calculi share a lot of material, which is why we present them in parallel, factorising what can be factorised: For instance, the syntaxes of the three calculi are fragments of a common grammar, for which a generic intersection type system specifies a notion of typing for each fragment. However, the calculi do not share the same reduction rules.

In this section, we first present the common grammar, then we define our generic intersection type system for it.

2.1 Terms

The syntaxes of the three calculi are subsets of a common grammar defined as follows:

$$M, N ::= \ x \mid \lambda x.M \mid MN \mid M[x := N] \mid W_x(M) \mid C_x^{y,z}(M)$$

The free variables $fv(M)$ of a term M are defined by the rules of figure 1.

$$fv(x) = \{x\} \qquad fv(\lambda x.M) = fv(M) - \{x\} \qquad fv(MN) = fv(M) \cup fv(N)$$

$$fv(M[x := N]) = (fv(M) - \{x\}) \cup fv(N) \qquad fv(W_x(M)) = fv(M) \cup \{x\}$$

$$fv(C_x^{y,z}(M)) = \begin{cases} (fv(M) - \{y, z\}) \cup \{x\} \text{ if } y \in fv(M) \text{ or } z \in fv(M) \\ fv(M) \text{ otherwise} \end{cases}$$

Figure 1: Free variables of a term

We consider terms up to α-equivalence and use Barendregt's convention [Bar84] to avoid variable capture.

Definition 1 (Linear terms)

- x is linear.
- If M and N are linear and $fv(M) \cap fv(N) = \varnothing$, then MN is linear.
- If M is linear and $x \in fv(M)$, then $\lambda x.M$ is linear.
- If M and N are linear, $x \in fv(M)$ and $(fv(M) - \{x\}) \cap fv(N) = \varnothing$, then $M[x := N]$ is linear.
- If M is linear and $x \notin fv(M)$, then $W_x(M)$ is linear.
- If M is linear, $y \in fv(M)$, $z \in fv(M)$, and $x \notin fv(M)$, then $C_x^{y,z}(M)$ is linear.

※

In this paper we consider in particular the three following fragments of the above syntax.

Definition 2 (Fragments)

Pure λ-calculus A λ-term is a term M that does not contain $M[x := N]$, or $W_x(M)$ or $C_x^{y,z}(M)$.

λS-calculus A λS-term is a term that does not contain $W_x(M)$ or $C_x^{y,z}(M)$.

λlxr-calculus A λlxr-term is a term M that is linear: every free and bound variable appears once and only once in the term (see [KL07]).

※

2.2 Types

To define the intersection type system, we first define the intersection types, which are the same for the three calculi.

Definition 3 (Types)

We consider a countable infinite set of elements called *atomic types* and use the type variable τ to range over it.

Intersection types are defined by the following syntax:

$$\begin{array}{lcll} F, G, \ldots & ::= & \tau \mid A \to F & F\text{-types} \\ A, B, \ldots & ::= & F \mid A \cap B & A\text{-types} \\ U, V, \ldots & ::= & A \mid \omega & U\text{-types} \end{array}$$

F-types are types that are not intersections, A-types are types that are not empty and U-types are types that can be empty.

We extend the intersection construct as an operation on U-types as follows:

$$A \cap \omega := A \quad \omega \cap A := A \quad \omega \cap \omega := \omega$$

※

Remark 1 For all U and V we have :

- $U \cap V$ exists.
- $U \cap \omega = \omega \cap U$
- If $U \cap V = \omega$, then $U = V = \omega$.

Note that we do **not** assume any implicit equivalence between intersection types (such as idempotency, associativity, commutativity).

F-types are similar to strict types defined in [vB92].

However, in order to prove theorems such as subject reduction we will need associativity and commutativity of the intersection $\cap$. So we define an equivalence relation on types.

Definition 4 ($\approx$) We inductively define $U \approx V$ by the rules of Fig. 2. ※

$$\frac{}{F \approx F} \qquad \frac{}{A \cap B \approx B \cap A} \qquad \frac{A \approx A' \quad B \approx B'}{A \cap B \approx A' \cap B'} \qquad \frac{A \approx B \quad B \approx C}{A \approx C}$$

$$\frac{}{(A \cap B) \cap C \approx A \cap (B \cap C)} \qquad \frac{}{A \cap (B \cap C) \approx (A \cap B) \cap C} \qquad \frac{}{\omega \approx \omega}$$

Figure 2: Equivalence on intersection types

The intersection types that we use here differ from those of [BL11a], in that the associativity and commutativity (AC) of the intersection $\cap$ are only featured "on the surface" of types, and not underneath functional arrows $\to$. This will make the typing rules much more syntax-directed, simplifying the proofs of soundness and completeness of typing with respect to the strong normalisation property. More to the point, this approach reduces the use of the AC properties to the only places where they are needed.

Lemma 2 (Properties of $\approx$) For all U, V, W, F, U', V',

1. $\approx$ is an equivalence relation.
2. If $U \approx \omega$, then $U = \omega$ and if $U \approx F$, then $U = F$.
3. $U \cap V \approx V \cap U$ and $(U \cap V) \cap W \approx U \cap (V \cap W)$.
4. If $U \approx U'$ and $V \approx V'$, then $U \cap V \approx U' \cap V'$.
5. For all U and V, if $U \cap V \approx U$, then $V = \omega$.

Proof: See Appendix C. □

We equip intersection types with a notion of sub-typing:

Definition 5 ($\subseteq$) We write $U \subseteq V$ if there exists U' such that $U \approx V \cap U'$. ※

Lemma 3 (Properties of $\subseteq$) For all U, U', V, V' :

1. $\subseteq$ is a partial pre-order for intersection types, and $U \approx U'$ if and only if $U \subseteq U'$ and $U' \subseteq U$.
2. $U \cap V \subseteq U$ and $U \subseteq \omega$
3. If $U \subseteq U'$ and $V \subseteq V'$, then $U \cap V \subseteq U' \cap V'$

Proof: Straightforward with Lemma 2. □

2.3 Typing contexts

We now lift those concepts to typing contexts before presenting the typing rules.

Definition 6 (Contexts)

A context Γ is a total map from variables to U-types such that $\mathsf{Dom}(\Gamma) := \{x \mid \Gamma(x) \neq \omega\}$ is finite. The intersection of contexts $\Gamma \cap \Delta$, the relations $\Gamma \approx \Delta$ and $\Gamma \subseteq \Delta$, are defined point-wise.

By $()$ we denote the context mapping every variable to ω and by $x{:}U$ the context mapping x to U and every other variable to ω.

The special case of $\Gamma \cap \Delta$ when $\mathsf{Dom}(\Gamma)$ and $\mathsf{Dom}(\Delta)$ are disjoint is denoted Γ, Δ. ※

Lemma 4 (Properties of contexts) For all contexts Γ, Γ', Δ, Δ', Γ'',

1. $\Gamma \cap () = \Gamma = () \cap \Gamma$ (for instance $\Gamma, x:\omega = \Gamma = x:\omega, \Gamma$)
2. If $\Gamma \cap \Delta = ()$, then $\Gamma = \Delta = ()$ and if $\Gamma \approx ()$, then $\Gamma = ()$
3. $\approx$ is an equivalence relation on contexts.
4. $\Gamma \cap \Delta \approx \Delta \cap \Gamma$ and $(\Gamma \cap \Gamma') \cap \Gamma'' \approx \Gamma \cap (\Gamma' \cap \Gamma'')$
5. If $\Gamma \approx \Gamma'$ and $\Delta \approx \Delta'$, then $\Gamma \cap \Gamma' \approx \Delta \cap \Delta'$
6. $\Gamma \subseteq \Delta$ if and only if there exists Γ' such that $\Gamma \approx \Delta \cap \Gamma'$.
7. $\subseteq$ is a partial pre-order for contexts, and $\Gamma \approx \Delta$ if and only if $\Gamma \subseteq \Delta$ and $\Delta \subseteq \Gamma$.
8. $\Gamma \cap \Delta \subseteq \Gamma$
9. If $\Gamma \subseteq \Gamma'$ and $\Delta \subseteq \Delta'$, then $\Gamma \cap \Delta \subseteq \Gamma' \cap \Delta'$.
10. $(\Gamma, x:U) \subseteq \Gamma$, in particular $\Gamma \subseteq ()$.

Proof: The proofs of these properties are straightforward with the use of Lemma 2 and Lemma 3. □

2.4 Typing judgements

Now that we have defined types and contexts we can define typing derivations. Instead of defining three typing systems for the three calculi we can define one typing system for the common grammar.

Definition 7 (Typability in System $\lambda_{\cap}^{\subseteq}$)
The judgement $\Gamma \vdash_{\cap\subseteq} M:U$ denotes the derivability of $\Gamma \vdash_{\cap\subseteq} M:U$ with the rules of Fig. 3. We write $\Gamma \vdash^{n}_{\cap\subseteq} M:U$ if there exists a derivation with n uses of the (App) rule. ※

$$\frac{}{x:F \vdash_{\cap\subseteq} x:F}\ (\text{Var}) \qquad \frac{\Gamma, x:U \vdash_{\cap\subseteq} M:F \quad A \subseteq U}{\Gamma \vdash_{\cap\subseteq} \lambda x.M : A \to F}\ (\text{Abs}) \qquad \frac{\Gamma \vdash_{\cap\subseteq} M:A \to F \quad \Delta \vdash_{\cap\subseteq} N:A}{\Gamma \cap \Delta \vdash_{\cap\subseteq} MN:F}\ (\text{App})$$

$$\frac{\Gamma \vdash_{\cap\subseteq} M:A \quad \Delta \vdash_{\cap\subseteq} M:B}{\Gamma \cap \Delta \vdash_{\cap\subseteq} M:A \cap B}\ (\text{Inter}) \qquad \frac{}{\vdash_{\cap\subseteq} M:\omega}\ (\text{Omega})$$

$$\frac{\Gamma \vdash_{\cap\subseteq} N:A \quad \Delta, x:U \vdash_{\cap\subseteq} M:F \quad U = A \vee U = \omega}{\Gamma \cap \Delta \vdash_{\cap\subseteq} M[x := N]:F}\ (\text{Subst})$$

$$\frac{\Gamma, x:U, y:V_1, z:V_2 \vdash_{\cap\subseteq} M:F}{\Gamma, x:U \cap (V_1 \cap V_2) \vdash_{\cap\subseteq} C_x^{y,z}(M):F}\ (\text{Contraction}) \qquad \frac{\Gamma, x:U \vdash_{\cap\subseteq} M:F}{\Gamma, x:U \cap A \vdash_{\cap\subseteq} W_x(M):F}\ (\text{Weakening})$$

Figure 3: System $\lambda_{\cap}^{\subseteq}$

Note that the rule deriving $\vdash_{\cap\subseteq} M:\omega$ does not interfere with the rest of the system as ω is not an A-type. It is only here for convenience to synthetically express some statements and proofs that would otherwise need a verbose case analysis (e.g. Lemma 8).

Examples of how λ-terms are typed are given in the next section.

Also, note that the introduction rule for the intersection is directed by the syntax of the types: If $\Gamma \cap \Delta \vdash_{\cap\subseteq} M:A \cap B$, then the last rule of the derivation is necessarily $(Inter)$ and its premises are necessarily $\Gamma \vdash_{\cap\subseteq} M:A$ and $\Delta \vdash_{\cap\subseteq} M:B$. We are not aware of any intersection type system featuring this property, which is here a consequence of dropping the implicit AC properties of intersections, and a clear advantage over the system in [BL11a]. Similar properties can however be found in systems such as those of [GILL11], which avoids having to explicitly type a term by an intersection type.

Lemma 5 (Basic properties of $\lambda^{\subseteq}_{\cap}$)

1. If $\Gamma \vdash^n_{\cap\subseteq} M : U \cap V$, then there exist Γ_1, Γ_2, n_1, n_2 such that $n = n_1 + n_2$, $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash^{n_1}_{\cap\subseteq} M : U$ and $\Gamma_2 \vdash^{n_2}_{\cap\subseteq} M : V$.
2. If $\Gamma \vdash_{\cap\subseteq} M : A$, then $\mathsf{Dom}(\Gamma) = fv(M)$.
3. If $\Gamma \vdash^n_{\cap\subseteq} M : U$ and $U \approx U'$, then there exists Γ' such that $\Gamma \approx \Gamma'$ and $\Gamma' \vdash^n_{\cap\subseteq} M : U'$
4. If $\Gamma \vdash^n_{\cap\subseteq} M : U$ and $U \subseteq V$, then there exist m and Δ such that $m \leq n$, $\Gamma \subseteq \Delta$ and $\Delta \vdash^m_{\cap\subseteq} M : V$

Proof: The first point generalises to U-types the previous remark. The second point is by induction on the typing tree. The third point is by induction on the derivation of $U \approx U'$, and the fourth one combines the previous points. □

The following lemma is used to prove Subject Reduction in both λS and λlxr:

Lemma 6 (Typing of explicit substitution) Assume $\Gamma, x : A \vdash^n_{\cap\subseteq} M : B$ and $\Delta \vdash^m_{\cap\subseteq} N : A$. Then, there exists Γ' such that $\Gamma' \approx \Gamma \cap \Delta$ and $\Gamma' \vdash^{n+m}_{\cap\subseteq} M[x := N] : B$.

Proof: See Appendix C. □

Remark 7 The previous theorem is not true if we replace A by ω: If B is an intersection, we would need to duplicate the typing tree of N.

3 Soundness

In this section we prove Subject Reduction and Soundness: respectively, the property that typing is preserved by reduction, and the property that if a term is typable, then it is strongly normalising for the reduction relation. This is true for the three calculi, but specific to each of them because the reduction relation is itself specific to each calculus.

Therefore in this section, we work separately on each calculus: each time, we define the reduction rules and prove the Subject Reduction property, which leads to Soundness.

3.1 Pure λ-calculus

Remember that a pure λ-term is a term M that does not contain any explicit substitutions $M[x := N]$, or weakenings $W_x(M)$ or contractions $C^{y,z}_x(M)$.

As we will see, only strongly normalising terms can be assigned an A-type by the system (Theorem 10). In fact, all of them can (Theorem 53), see for instance how the example below correctly uses the abstraction rule ($A \subseteq \omega$).

$$\dfrac{\dfrac{}{x : F, y : \omega \vdash_{\cap\subseteq} x : F}}{x : F \vdash_{\cap\subseteq} \lambda y.x : A \to F}$$

Owing to non-idempotency, no closed term inhabits the simple type $(\tau \to \tau \to \tau') \to (\tau \to \tau')$ (with $\tau \neq \tau'$), but its natural inhabitant $\lambda f.\lambda x.f\ x\ x$ in a simply-typed system can here be given type $(\tau \to \tau \to \tau') \to (\tau \cap \tau \to \tau')$.

Definition 8 (Reduction in λ-calculus)
If M and N are pure λ-terms, we denote by $M\{x := N\}$ the result of the (implicit) substitution (as defined in e.g. [Bar84]).

The reduction rule is β-reduction:

$$(\lambda x.M)\ N \longrightarrow M\{x := N\}$$

The congruent closure of this rule is denoted $\longrightarrow_\beta$. SN^λ denotes the set of strongly normalising λ-terms (for β-reduction).

※

Lemma 8 (Typing of implicit substitutions) If $\Gamma, x : U \vdash^{n}_{\cap\subseteq} M : A$ and $\Delta \vdash^{m}_{\cap\subseteq} N : U$, then there exists Γ' such that $\Gamma' \approx \Gamma \cap \Delta$ and $\Gamma' \vdash^{n+m}_{\cap\subseteq} M\{x := N\} : A$.

Proof: See Appendix C. □

Theorem 9 (Subject Reduction for λ) If $\Gamma \vdash^{n}_{\cap\subseteq} M : A$ and $M \longrightarrow_{\beta} M'$, then there exist m and Δ such that $m < n$, $\Gamma \subseteq \Delta$ and $\Delta \vdash^{m}_{\cap\subseteq} M' : A$.

Proof: See Appendix C. □

The above theorem and its proof are standard but for the quantitative information in the typability properties. This is where non-idempotent intersections provide a real advantage over idempotent ones, as every β-reduction strictly reduces the number of application rules in the typing trees: no sub-tree is duplicated in the process. This is something specific to non-idempotent intersection types, and obviously false for simple types or idempotent intersection types.

As a direct corollary we obtain:

Theorem 10 (Soundness for λ) If M is a pure λ-term and $\Gamma \vdash_{\cap\subseteq} M : A$, then $M \in \mathsf{SN}^{\lambda}$.

The converse is also true (strongly normalising terms can be typed in $\lambda^{\subseteq}_{\cap}$), see Theorem 53 and more generally section 5.3 (with Subject Expansion, etc.).

3.2 λS

Remember that terms of λS are terms that do not contain any weakenings $W_x(M)$ or contractions $C^{y,z}_{x}(M)$. In other words, we consider the extension of the pure λ-calculus with explicit substitutions $(M[x := N])$. This is the same syntax as that of $\lambda\mathsf{x}$ [BR95], but unlike $\lambda\mathsf{x}$, the reduction rules only duplicate substitutions when needed. For example, the following rule:

$$(M_1 M_2)[x := N] \longrightarrow M_1[x := N]M_2[x := N]$$

can only be applied if $x \in fv(M_1)$ and $x \in fv(M_2)$.

In the other cases, the explicit substitution will only go one way. The rules are chosen with the proof of Subject Reduction in mind, which is a simple adaptation of the proof in the pure λ-calculus. This leads to soundness (typable implies strongly normalising), and therefore Melliès's counter-example [Mel95] to strong normalisation is naturally avoided.

Definition 9 (Reduction in λS)

The reduction and equivalence rules of λS are presented in Fig. 4.

For a set of rules $E \subseteq \{B, S, W\}$ from Figure 4, $\longrightarrow_E$ denotes the congruent closure of the rules in E modulo the $\equiv$ rule.

$\mathsf{SN}^{\lambda S}$ denotes the set of strongly normalising λS-terms for $\longrightarrow_{B,S,W}$. ※

We call this calculus λS because it is a variant of the calculi λ_s of [Kes07] and λ_{es} of [Ren11]. That of [Ren11] is more general than that of [Kes07] in the sense that it allows the reductions

$$\begin{array}{lllll} (1) & (M_1 M_2)[x := N] & \longrightarrow & M_1[x := N]M_2 & \text{when } x \notin fv(M_1), x \notin fv(M_2) \\ (2) & (M_1 M_2)[x := N] & \longrightarrow & M_1 M_2[x := N] & \text{when } x \notin fv(M_1), x \notin fv(M_2) \end{array}$$

Reduction (2) is problematic in our approach since, even though the Subject Reduction property would still hold, it would not hold with the quantitative information from which Strong Normalisation can be proved: In the typing tree, the type of M_1 is not an intersection (it is an F-type) but the type of M_2 can be one. So we cannot directly type $M_2[x := N]$. If $x \in fv(M_2)$ we can use Lemma 6, otherwise we have to duplicate the typing tree of N.

We therefore exclude (2) from the calculus, but keep (1) as one of our rules, since it is perfectly compatible with our approach. It is also needed to simulate (in several steps) the general *garbage collection* rule below

$$M[x := N] \longrightarrow M \quad (x \notin fv(M))$$

$B:$	$(\lambda x.M)N$	$\longrightarrow$	$M[x := N]$	
$W:$	$y[x := N]$	$\longrightarrow$	y	$x \neq y$
$S:$				
	$x[x := N]$	$\longrightarrow$	N	(SR)
	$(M_1M_2)[x := N]$	$\longrightarrow$	$(M_1[x := N])(M_2[x := N])$	$x \in fv(M_1), x \in fv(M_2)$
	$(M_1M_2)[x := N]$	$\longrightarrow$	$M_1(M_2[x := N])$	$x \notin fv(M_1), x \in fv(M_2)$
	$(M_1M_2)[x := N]$	$\longrightarrow$	$(M_1[x := N])M_2$	$x \notin fv(M_2)$
	$(\lambda y.M)[x := N]$	$\longrightarrow$	$\lambda y.M[x := N]$	$x \neq y, y \notin fv(N)$
	$(M_1[y := M_2])[x := N]$	$\longrightarrow$	$(M_1[x := N])[y := M_2[x := N]]$	$x \in fv(M_1), x \in fv(M_2), y \notin fv(N)$
	$(M_1[y := M_2])[x := N]$	$\longrightarrow$	$M_1[y := M_2[x := N]]$	$x \notin fv(M_1), x \in fv(M_2)$
	$(M[x := N_1])[y := N_2]$	$\equiv$	$(M[y := N_2])[x := N_1]$	$x \neq y, x \notin fv(N_2), y \notin fv(N_1)$

Figure 4: Reduction and equivalence rules of λS

which is present in both [Kes07] and [Ren11], and which we decide to restrict, for simplicity, to the case where M is a variable different from x.[5] All of our results would still hold with the general garbage collection rule.

Lemma 11 $\longrightarrow_{S,W}$ terminates.

Proof: By a polynomial argument. More precisely, see appendix C. □

Theorem 12 (Subject Reduction for λS)
Assume $\Gamma \vdash^n_{\cap\subseteq} M : A$. We have the following properties:

- If $M \longrightarrow_B M'$, then there exist Γ' and m such that $\Gamma \subseteq \Gamma'$, $m < n$ and $\Gamma' \vdash^m_{\cap\subseteq} M' : A$
- If $M \longrightarrow_S M'$, then there exists Γ' such that $\Gamma \approx \Gamma'$ and $\Gamma' \vdash^n_{\cap\subseteq} M' : A$
- If $M \longrightarrow_W M'$, then there exist Γ' and m such that $\Gamma \subseteq \Gamma'$, $m \leq n$ and $\Gamma' \vdash^m_{\cap\subseteq} M' : A$
- If $M \equiv M'$, then there exists Γ' such that $\Gamma \approx \Gamma'$ and $\Gamma' \vdash^n_{\cap\subseteq} M' : A$

Proof: See Appendix C. □

Theorem 13 (Soundness for λS) If M is a λS-term and $\Gamma \vdash_{\cap\subseteq} M : A$, then $M \in \mathsf{SN}^{\lambda S}$.

Proof: We have $\Gamma \vdash^n_{\cap\subseteq} M : A$ for some n, and strong normalisation is provided by a lexicogrphic argument as follows:

- $\longrightarrow_B$ strictly decreases n
- $\longrightarrow_S$ and $\longrightarrow_W$ decrease n or do not change it
- $\longrightarrow_{S,W}$ terminates on its own.

□

3.3 λlxr

Remember that λlxr-terms are terms that are linear. In particular the typing rules of abstraction, explicit substitution, weakening and contraction, degenerate into the following rules when linearity is assumed:

[5] [Kes07] needs the general version, if only for the lack of rule (1).

$$\frac{A \subseteq C \quad \Gamma, x{:}C \vdash_{\cap\subseteq} M{:}F}{\Gamma \vdash_{\cap\subseteq} \lambda x.M{:}A \to F} \qquad \frac{\Gamma \vdash_{\cap\subseteq} N{:}B \quad \Delta, x{:}B \vdash_{\cap\subseteq} M{:}F}{\Gamma \cap \Delta \vdash_{\cap\subseteq} M[x := N]{:}F}$$

$$\frac{\Gamma \vdash_{\cap\subseteq} M{:}F}{\Gamma, x{:}A \vdash_{\cap\subseteq} W_x(M){:}F} \qquad \frac{\Gamma, y{:}A, z{:}B \vdash_{\cap\subseteq} M{:}F}{\Gamma, x{:}A \cap B \vdash_{\cap\subseteq} C_x^{y,z}(M){:}F}$$

Remark 14 Had we defined the ($Weakening$) and ($Contraction$) rules as above from the start (cf. Fig. 3), we would have had to add extra conditions for Theorems 21.5 and 21.6 to hold. This would amount to assuming some of the consequences of linearity. We prefer to keep Theorem 21 and the whole of Section 4 calculus-independent, by giving a more general definition of the two rules.

Definition 10 (Reduction in λlxr)

The reduction and equivalence rules of λlxr are presented in Fig. 5.

For a set of rules E from Fig. 5 denotes the congruent closure of the rules in E modulo the equivalence rules.

$\mathsf{SN}^{\lambda lxr}$ denotes the set of strongly normalising λlxr-terms for the entire reduction relation.

※

$B:$	$(\lambda x.M)N$	$\longrightarrow$	$M[x := N]$	
$SR:$	$x[x := N]$	$\longrightarrow$	N	
$SP:$	$(M_1M_2)[x := N]$	$\longrightarrow$	$(M_1[x := N])M_2$	$x \in fv(M_1)$
	$(M_1M_2)[x := N]$	$\longrightarrow$	$M_1(M_2[x := N])$	$x \in fv(M_2)$
	$(\lambda y.M)[x := N]$	$\longrightarrow$	$\lambda y.M[x := N]$	$x \neq y, y \notin fv(N)$
	$W_y(M)[x := N]$	$\longrightarrow$	$W_y(M[x := N])$	$x \neq y$
	$C_y^{z1,z2}(M)[x := N]$	$\longrightarrow$	$C_y^{z1,z2}(M[x := N])$	$y \neq x, y \notin fv(N), z_i \notin fv(N)$
$W:$	$W_x(M)[x := N]$	$\longrightarrow$	$W_{fv(N)}(M)$	
$D:$	$C_x^{y,z}(M)[x := N]$	$\longrightarrow$	$C_X^{Y,Z}(M[y := N_1][z := N_2])$	
$ACC:$	$C_w^{x,v}(C_x^{z,y}(M))$	$\equiv$	$C_w^{x,y}(C_x^{z,v}(M))$	$x \neq y, v$
	$C_x^{y,z}(M)$	$\equiv$	$C_x^{z,y}(M)$	
	$C_{x'}^{y',z'}(C_x^{y,z}(M))$	$\equiv$	$C_x^{y,z}(C_{x'}^{y',z'}(M))$	$x \neq y', z' \& x' \neq y, z$
	$M_1[y := M_2][x := N]$	$\longrightarrow$	$M_1[y := M_2[x := N]]$	$x \in fv(M_2)$
$ACW:$	$W_x(W_y(M))$	$\equiv$	$W_y(W_x(M))$	
$CS:$	$M[x := N_1][y := N_2]$	$\equiv$	$M[y := N_2][x := N_1]$	$y \notin fv(N_1), x \notin fv(N_2)$
	$C_w^{y,z}(M)[x := N]$	$\equiv$	$C_w^{y,z}(M[x := N])$	$x \neq w, \& y, z \notin fv(N)$
$WAbs:$	$\lambda x.W_y(M)$	$\longrightarrow$	$W_y(\lambda x.M)$	$x \neq y$
$WApp1:$	$W_y(M)N$	$\longrightarrow$	$W_y(MN)$	
$WApp2:$	$MW_y(N)$	$\longrightarrow$	$W_y(MN)$	
$WSubs:$	$M[x := W_y(N)]$	$\longrightarrow$	$W_y(M[x := N])$	
$Merge:$	$C_w^{y,z}(W_y(M))$	$\longrightarrow$	$R_w^z(M)$	
$Cross:$	$C_w^{y,z}(W_x(M))$	$\longrightarrow$	$W_x(C_w^{y,z}(M))$	$x \neq y, x \neq z$
$CAbs:$	$C_w^{y,z}(\lambda x.M)$	$\longrightarrow$	$\lambda x.C_w^{y,z}(M)$	
$CApp1:$	$C_w^{y,z}(MN)$	$\longrightarrow$	$C_w^{y,z}(M)N$	$y, z \in fv(M)$
$CApp2:$	$C_w^{y,z}(MN)$	$\longrightarrow$	$MC_w^{y,z}(N)$	$y, z \in fv(N)$
$CSubs:$	$C_w^{y,z}(M[x := N])$	$\longrightarrow$	$M[x := C_w^{y,z}(N)]$	$y, z \in fv(N)$

In rule D, $X = fv(N)$, $N_1 = N\{\vec{X} := \vec{Y}\}$, $N_2 = N\{\vec{X} := \vec{Z}\}$, Y, Z fresh sets of variables in bijection with X. In rule $Merge$, $R_w^z(M)$ is the renaming of z by w in M.

Figure 5: Reduction and equivalence rules of λlxr

Theorem 15 (Subject Reduction for λlxr) If $\Gamma \vdash^n_{\cap\subseteq} M : A$ then:

- If $M \longrightarrow_B M'$, then there exist Γ' and m such that $\Gamma \subseteq \Gamma'$, $m < n$, and $\Gamma' \vdash^m_{\cap\subseteq} M' : A$
- If $M \longrightarrow_E M'$ and $B \notin E$, then there exist Γ' and m such that $\Gamma \subseteq \Gamma'$, $m \leq n$ and $\Gamma' \vdash^m_{\cap\subseteq} M' : A$.
- If $M \equiv M'$ then there exist Γ' such that $\Gamma \approx \Gamma'$ and $\Gamma' \vdash^n_{\cap\subseteq} M' : A$.

Proof: See Appendix C. □

Theorem 16 (Soundness for λlxr) If M is a λlxr-term and $\Gamma \vdash_{\cap\subseteq} M : A$, then $M \in \mathsf{SN}^{\lambda lxr}$.

Proof: Similarly to λS we have $\Gamma \vdash^n_{\cap\subseteq} M : A$ for some n, and strong normalisation is provided by a lexicogrphic argument as follows:

- $\longrightarrow_B$ strictly decreases n
- Any other reduction $\longrightarrow_E$ decreases n or does not change it
- The reduction system without the B rule terminates on its own [KL07].

□

4 Denotational semantics for strong normalisation

In this section we show how to use non-idempotent intersection types to simplify the methodology of [CS07], which we briefly review here:

The goal is to produce modular proofs of strong normalisation for various *source typing systems*. The problem is reduced to the strong normalisation of a unique *target system* of intersection types, chosen once and for all. This is done by interpreting each term t as the set $[\![t]\!]$ of the intersection types that can be assigned to t in the target system. Two facts then remain to be proved:

1. if t can be typed in the source system, then $[\![t]\!]$ is not empty
2. the target system is strongly normalising

The first point is the only part that is specific to the source typing system: it amounts to turning the interpretation of terms into a filter model of the source typing system. The second point depends on the chosen target system: as [CS07] uses a system of *idempotent* intersection types (extending the simply-typed λ-calculus), their proof involves the usual reducibility technique [Gir72, Tai75]. But this is somewhat redundant with point 1 which uses similar techniques to prove the correctness of the filter model with respect to the source system.[6]

In this paper we propose to use *non-idempotent* intersection types for the target system, so that point 2 can be proved with simpler techniques than in [CS07] while point 1 is not impacted by the move. In practice we propose $\lambda^{\subseteq}_{\cap}$ as the target system (that of [BL11a] would work just as well). The present section shows the details of this alternative.

Notice that $\lambda^{\subseteq}_{\cap}$ is not an extension of the simply-typed λ-calculus, in that a typing tree in the system of simple types is not a valid typing tree in system $\lambda^{\subseteq}_{\cap}$, which uses non-idempotent intersections (while it is a valid typing tree in the system of [CS07] which uses idempotent intersections). But a nice application of our proposed methodology is that, by taking the simply-typed lambda-calculus as the source system, we can produce a typing tree in $\lambda^{\subseteq}_{\cap}$ from a typing tree with simple types. We do not know of any more direct encoding.

[6] If reducibility techniques are needed for the latter, why not use them on the source system directly (besides formulating a modular methodology)?

4.1 I-filters

The following filter constructions only involve the syntax of types and are independent from the chosen target system.

Definition 11 (I-filter)

- An I-filter is a set v of A-types such that:
  - for all A and B in v we have $A \cap B \in v$
  - for all A and B, if $A \in v$ and $A \subseteq B$, then $B \in v$
- In particular the empty set and the sets of all A-types are I-filters and we write them $\bot$ and $\top$ respectively.
- Let $\mathcal{D}$ be the sets of all non-empty I-filters; we call such I-filters *values*.
- Let $\mathcal{E}$ be the sets of all I-filters ($\mathcal{E} = \mathcal{D} \cup \{\bot\}$).

※

While our intersection types differ from those in [CS07] (in that idempotency is dropped), the stability of a filter under type intersections makes it validate idempotency (it contains A if and only if it contains $A \cap A$, etc). This makes our filters very similar to those in [CS07], so we can plug-in the rest of the methodology with minimal change.

Remark 17 (Basic properties of I-filters)

1. If $(v_i)_{i\in I}$ is an non empty family of $\mathcal{E}$, then $\bigcap_{i\in I} v_i \in \mathcal{E}$.
2. If v is a set of A-types, then there is a smallest $v' \in \mathcal{E}$ such that $v \subseteq v'$ and we write $< v >:= v'$.
3. If v is a set of F-types, then $< v >$ is the closure of v under finite intersections.
4. If $v \in \mathcal{E}$, then $v =< \{F \mid F \in v\} >$.
5. If u and v are sets of F-types such that $< u >=< v >$, then $u = v$.

Hence, in order to prove that two I-filters are equal we just have to prove that they contain the same F-types.

I-filters form an applicative structure:

Definition 12 (Application of I-filters) If u, v are in $\mathcal{E}$, then define

$$u@v := < \{F \mid \exists A \in v, (A \to F) \in u\} >$$

※

Remark 18 For all $u \in \mathcal{E}$, $u@\bot = \bot@u = \bot$, and for all $u \in \mathcal{D}$, $\top@u = \top$.

Definition 13 (Environments and contexts)

An *environment* is a map from term variables $x, y, \ldots$ to I-filters.

If ρ is an environment and Γ is a context, we say that $\Gamma \in \rho$, or Γ is *compatible with* ρ, if for all x, $\Gamma(x) = \omega$ or $\Gamma(x) \in \rho(x)$.

Assume ρ is an environment, u is an I-filter and x is a variable. Then, the environment $\rho, x \mapsto u$ is defined as follows:

$$\begin{array}{lcll} (\rho, x \mapsto u)(x) & := & u & \\ (\rho, x \mapsto u)(y) & := & \rho(y) & \forall y \neq x \end{array}$$

※

Remark 19 (Environments are I-filters of contexts) [7] Let ρ be an environment.

1. If $\Gamma \in \rho$ and $\Gamma' \in \rho$, then $\Gamma \cap \Gamma' \in \rho$.
2. If $\Gamma \in \rho$ and Γ' is a context such that $\Gamma \subseteq \Gamma'$, then $\Gamma' \in \rho$.

4.2 Semantics of terms as I-filters

The remaining ingredients now involve the target system; we treat here $\lambda^{\subseteq}_{\cap}$.

Definition 14 (Interpretation of terms)
If M is a term and ρ is an environment we define

$$[\![M]\!]_\rho := \{A \mid \exists \Gamma \in \rho, \Gamma \vdash_{\cap\subseteq} M : A\}$$

※

Remark 20 $[\![M]\!]_\rho \in \mathcal{E}$, and therefore $[\![M]\!]_\rho =< \{F \mid \exists \Gamma \in \rho, \Gamma \vdash_{\cap\subseteq} M : F\} >$.

Theorem 21 (Inductive characterisation of the interpretation)

1. $[\![x]\!]_\rho = \rho(x)$
2. $[\![MN]\!]_\rho = [\![M]\!]_\rho @ [\![N]\!]_\rho$
3. $[\![\lambda x.M]\!]_\rho @ u = [\![M]\!]_{\rho, x\mapsto u}$ if $u \neq \bot$.
4. $[\![M[x := N]]\!]_\rho = [\![M]\!]_{\rho, x\mapsto [\![N]\!]_\rho}$ if $[\![N]\!]_\rho \neq \bot$
5. $[\![W_x(M)]\!]_\rho = [\![M]\!]_\rho$ if $\rho(x) \neq \bot$.
6. $[\![C^{y,z}_x(M)]\!]_\rho = [\![M]\!]_{\rho, y\mapsto\rho(x), z\mapsto\rho(x)}$.

Proof: See Appendix C. □

This theorem makes $\lambda^{\subseteq}_{\cap}$ a suitable alternative as a target system: the filter models of the source systems treated in [CS07] can be done with a system of non-idempotent intersection types. While we could develop those constructions, we prefer to cover a new range of source systems: those with second-order quantifiers such as System F.

[7] Conversely, if E is an I-filter of contexts, then ρ, defined by $\rho(x) = \{\Gamma(x) \neq \omega \mid \Gamma \in E\}$ for all x, is an environment.

4.3 An example: System F and the likes

Definition 15 (Types and Typing System) Types are built by the following grammar:

$$\mathfrak{A}, \mathfrak{B}, \ldots ::= \alpha \mid \mathfrak{A} \to \mathfrak{B} \mid \mathfrak{A} \cap \mathfrak{B} \mid \forall \alpha \mathfrak{A}$$

where α denotes a type variable, $\forall \alpha \mathfrak{A}$ binds α in $\mathfrak{A}$, types are considered modulo α-conversion, and $ftv(\mathfrak{A})$ denotes the free (type) variables of $\mathfrak{A}$.

Typing contexts, denoted $\mathfrak{G}, \mathfrak{H}, \ldots$ are partial maps from term variables to types, and $(x : \mathfrak{A})$ denotes the map from x to $\mathfrak{A}$.

Let $\mathcal{S}$ be the typing system consisting of the rules in Fig. 6.

Typability in system $\mathcal{S}$ will be expressed by judgements of the form $\mathfrak{G} \vdash_{\mathcal{S}} M : \mathfrak{A}$.

※

$$\frac{}{\mathfrak{G}, x:\mathfrak{A} \vdash_{\cap\subseteq} x:\mathfrak{A}} \qquad \frac{\mathfrak{G}, x:\mathfrak{A} \vdash_{\cap\subseteq} M:\mathfrak{B}}{\mathfrak{G} \vdash_{\cap\subseteq} \lambda x.M : \mathfrak{A}\to\mathfrak{B}} \qquad \frac{\mathfrak{G} \vdash_{\cap\subseteq} M:\mathfrak{A}\to\mathfrak{B} \quad \mathfrak{G} \vdash_{\cap\subseteq} N:\mathfrak{A}}{\mathfrak{G} \vdash_{\cap\subseteq} M\ N : \mathfrak{B}}$$

$$\frac{\mathfrak{G} \vdash_{\cap\subseteq} M:\mathfrak{A} \quad \mathfrak{G} \vdash_{\cap\subseteq} M:\mathfrak{B}}{\mathfrak{G} \vdash_{\cap\subseteq} M:\mathfrak{A}\cap\mathfrak{B}} \qquad \frac{\mathfrak{G} \vdash_{\cap\subseteq} M:\mathfrak{A}\cap\mathfrak{B}}{\mathfrak{G} \vdash_{\cap\subseteq} M:\mathfrak{A}} \qquad \frac{\mathfrak{G} \vdash_{\cap\subseteq} M:\mathfrak{A}\cap\mathfrak{B}}{\mathfrak{G} \vdash_{\cap\subseteq} M:\mathfrak{B}}$$

$$\frac{\mathfrak{G} \vdash_{\cap\subseteq} M:\mathfrak{A}}{\mathfrak{G} \vdash_{\cap\subseteq} M:\forall\alpha\mathfrak{A}}\ \alpha \notin ftv(\mathfrak{G}) \qquad \frac{\mathfrak{G} \vdash_{\cap\subseteq} M:\forall\alpha\mathfrak{A}}{\mathfrak{G} \vdash_{\cap\subseteq} M:\mathfrak{A}\{\alpha := \mathfrak{B}\}}$$

$$\frac{\mathfrak{G} \vdash_{\cap\subseteq} N:\mathfrak{A} \quad \mathfrak{G}, x:\mathfrak{A} \vdash_{\cap\subseteq} M:\mathfrak{B}}{\mathfrak{G} \vdash_{\cap\subseteq} M[x := N]:\mathfrak{B}} \qquad \frac{\mathfrak{G} \vdash_{\cap\subseteq} M:\mathfrak{B} \quad x \notin dom(\mathfrak{G})}{\mathfrak{G}, x:\mathfrak{A} \vdash_{\cap\subseteq} W_x(M):\mathfrak{B}} \qquad \frac{\mathfrak{G}, y:\mathfrak{A}, z:\mathfrak{A} \vdash_{\cap\subseteq} M:\mathfrak{B}}{\mathfrak{G}, x:\mathfrak{A} \vdash_{\cap\subseteq} C_x^{y,z}(M):\mathfrak{B}}$$

Figure 6: Miscellaneous Typing Rules

Remark 22 Note that the typing rules in System $\mathcal{S}$ do not necessarily follow the philosophy of the λlxr-calculus and the λS-calculus. For example, we would expect a typing system for λS or λlxr to be such that the domain of the typing context is exactly the set of free variables in the typed term (this leads to interesting properties and better encodings into proof-nets -see e.g. [KL07]).

However here, we are only interested in strong normalisation, and we therefore consider a typing system as general as possible (hence the accumulation of rules in Fig. 6), i.e. a typing system such that the terms that are typed in an appropriate typing system (such as that of [KL07] for λlxr) can be typed here. This is the case of System $\mathcal{S}$. Alternatively, we could also adapt and tailor the proof of strong normalisation below to the specific typing system in which we are interested.

4.4 An intuitionistic realisability model

We now build the model $\mathcal{M}^{\mathsf{i}}_{\mathcal{F}}$ as follows:

Definition 16 (Realisability Predicate) A *realisability predicate* is a subset X of $\mathcal{D}$ containing $\top$. We define $\mathsf{TP}(\mathcal{D})$ as the set of realisability predicates. ※

Lemma 23 (Shape of realisability predicates)

1. If $(X_i)_{i\in I}$ is a non empty family of $\mathsf{TP}(\mathcal{D})$, then $\bigcap_{i\in I} X_i \in \mathsf{TP}(\mathcal{D})$.
2. If X and Y in $\mathsf{TP}(\mathcal{D})$, then $X \to Y \in \mathsf{TP}(\mathcal{D})$ where $X \to Y$ is defined as
$$X \to Y := \{u \mid \forall v \in X, u@v \in Y\}$$

Proof: The only subtle point is the second one: First, for all $v \in X$, $v \neq \bot$ and thus $\top@v = \top \in Y$. So $\top \in X \to Y$. Second, suppose that $\bot \in X \to Y$. As $X \neq \emptyset$, there is $u \in X$, for which $\bot@u = \bot \in Y$, which contradicts the fact that $Y \in \mathsf{TP}(\mathcal{D})$. □

We can now interpret types:

Definition 17 (Interpretation of types)
Valuations are mappings from type variables to elements of $\mathsf{TP}(\mathcal{D})$.

Given such a valuation σ, the interpretation of types is defined as follows:

$$\begin{array}{llll} [\![\alpha]\!]_\sigma & := \sigma(\alpha) & [\![\mathfrak{A}\cap\mathfrak{B}]\!]_\sigma & := [\![\mathfrak{A}]\!]_\sigma \cap [\![\mathfrak{B}]\!]_\sigma \\ [\![\mathfrak{A}{\to}\mathfrak{B}]\!]_\sigma & := [\![\mathfrak{A}]\!]_\sigma{\to}[\![\mathfrak{B}]\!]_\sigma & [\![\forall\alpha\mathfrak{A}]\!]_\sigma & := \bigcap_{X\in\mathsf{TP}(\mathcal{D})} [\![\mathfrak{A}]\!]_{\sigma,\alpha\mapsto X} \end{array}$$

The interpretation of typing contexts is defined as follows:

$$[\![\mathfrak{G}]\!]_\sigma := \{\rho \mid \forall(x{:}\mathfrak{A})\in\mathfrak{G}, \rho(x)\in[\![\mathfrak{A}]\!]_\sigma\}$$

※

Finally we get Adequacy:

Lemma 24 (Adequacy Lemma) If $\mathfrak{G}\vdash_{\mathcal{S}} M:\mathfrak{A}$, then for all valuations σ and for all mappings $\rho\in[\![\mathfrak{G}]\!]_\sigma$ we have $[\![M]\!]_\rho\in[\![\mathfrak{A}]\!]_\sigma$.

Proof: By induction on the derivation of $\mathfrak{G}\vdash_{\mathcal{S}} M:\mathfrak{A}$, using Theorem 21 (and the fact that $[\![\mathfrak{A}\{\alpha:=\mathfrak{B}\}]\!]_\sigma = [\![\mathfrak{A}]\!]_{\sigma,\alpha\mapsto[\![\mathfrak{A}]\!]_\sigma}$, which is proved by induction on $\mathfrak{A}$). □

Corollary 25 (Strong normalisation of $\mathcal{S}$) If $\mathfrak{G}\vdash_{\mathcal{S}} M:\mathfrak{A}$, then $M\in\mathsf{SN}$.

Proof: Applying the previous lemma with σ mapping every type variable to $\{\top\}$ and ρ mapping all term variables to $\top$, we get $[\![M]\!]_\rho\in[\![\mathfrak{A}]\!]_\sigma$, so $[\![M]\!]_\rho\neq\bot$. Hence, M can be typed in $\lambda^{\subseteq}_{\cap}$, so $M\in\mathsf{SN}$. □

The advantage of non-idempotent intersection types (over idempotent ones) lies in the very last step of the above proof: here the typing trees of $\lambda^{\subseteq}_{\cap}$ get smaller with every β-reduction (proof of Theorem 10), while a reducibility technique as in [CS07] combines yet again an induction on types with an induction on typing trees similar to that in the Adequacy Lemma.

4.5 Orthogonality models

In this section we show how the above methodology can be integrated to the theory of *orthogonality*, i.e. how this kind of filter model construction can be captured by orthogonality techniques [Gir87, DK00, Kri01, MM09]. These techniques are particularly suitable to prove that typed terms satisfy some property [Par97, MV05, LM08], the most well-known of which being Strong Normalisation.

For this we define an abstract notion of orthogonality model for the system $\mathcal{S}$ defined in Fig. 6. In particular our definition also applies to sub-systems such as the simply-typed λ-calculus, the idempotent intersection type system, System F, etc. We could also adapt it with no difficulty to accommodate System F_ω.

Orthogonality techniques and the filter model construction from Section 4.1 (with the sets $\mathcal{D}$ and $\mathcal{E}$) inspire the notion of orthogonality model below. First we need the following notations:

Notation 18 Given a set $\mathcal{D}$, let $\mathcal{D}^*$ be the set of lists of elements of $\mathcal{D}$, with $[]$ representing the empty list and $u::\vec{v}$ representing the list of head u and tail $\vec{v}$.

Definition 19 (Orthogonality model)
An *orthogonality model* is a 4-tuple $(\mathcal{E},\mathcal{D},\ \perp\!\!\!\perp\ ,[\![\_]\!]_\_)$ where

- $\mathcal{E}$ is a set, called the *support*
- $\mathcal{D}\subseteq\mathcal{E}$ is a set of elements called *values*
- $\perp\!\!\!\perp\ \subseteq\mathcal{D}\times\mathcal{D}^*$ is called the *orthogonality relation*
- $[\![\_]\!]_\_$ is a function mapping every term M (typed or untyped) to an element $[\![M]\!]_\rho$ of the support, where ρ is a parameter called *environment* mapping term variables to values.

- the following axioms are satisfied:
  (A1) For all ρ, $\vec{v}$, x, if $\rho(x) \perp\!\!\!\perp \vec{v}$, then $[\![x]\!]_\rho \perp\!\!\!\perp \vec{v}$.
  (A2) For all ρ, $\vec{v}$, M_1, M_2, if $[\![M_1]\!]_\rho \perp\!\!\!\perp ([\![M_2]\!]_\rho :: \vec{v})$, then $[\![M_1\ M_2]\!]_\rho \perp\!\!\!\perp \vec{v}$.
  (A3) For all ρ, $\vec{v}$, x, M and for all values u,
  if $[\![M]\!]_{\rho,x\mapsto u} \perp\!\!\!\perp \vec{v}$, then $[\![\lambda x.M]\!]_\rho \perp\!\!\!\perp (u :: \vec{v})$.
  (A4) For all ρ, $\vec{v}$, x, M_1, M_2,
  if $[\![M_2]\!]_\rho$ is a value and $[\![M_1]\!]_{\rho,x\mapsto[\![M_2]\!]_\rho} \perp\!\!\!\perp \vec{v}$, then $[\![M_1[x := M_2]]\!]_\rho \perp\!\!\!\perp \vec{v}$.
  (A5) For all ρ, $\vec{v}$, x, M (such that $x \notin fv(M)$) and for all values u,
  if $[\![M]\!]_\rho \perp\!\!\!\perp \vec{v}$, then $[\![W_x(M)]\!]_{\rho,x\mapsto u} \perp\!\!\!\perp \vec{v}$.
  (A6) For all ρ, $\vec{v}$, x, y, z (distinct variables), M (such that $x \notin fv(M)$) and for all values u,
  if $[\![M]\!]_{\rho,y\mapsto u,z\mapsto u} \perp\!\!\!\perp \vec{v}$, then $[\![C_x^{y,z}(M)]\!]_{\rho,x\mapsto u} \perp\!\!\!\perp \vec{v}$.

※

In fact, $\mathcal{D}$ and $\perp\!\!\!\perp$ are already sufficient to interpret any type A as a set $[\![A]\!]$ of values: if types are seen as logical formulae, we can see this construction as a way of building some of their realisability / set-theoretical models.

There is no notion of computation pertaining to values, but the interplay between the interpretation of terms and the orthogonality relation is imposed by the axioms so that the Adequacy Lemma (which relates typing to semantics) holds:

$$\text{If } \vdash^{\mathcal{S}}_{\cap\subseteq} M : \mathfrak{A}, \text{ then } [\![M]\!] \in [\![\mathfrak{A}]\!]$$

4.5.1 Semantics of types and Adequacy Lemma

Definition 20 (Orthogonal)

- If $X \subseteq \mathcal{D}$, then let $X^\perp := \{\vec{v} \in \mathcal{D}^* \mid \forall u \in X, u \perp\!\!\!\perp \vec{v}\}$
- If $Y \subseteq \mathcal{D}^*$, then let $Y^\perp := \{u \in \mathcal{D} \mid \forall \vec{v} \in Y, u \perp\!\!\!\perp \vec{v}\}$

※

Remark 26 If $X \subseteq \mathcal{D}$ or $X \subseteq \mathcal{D}^*$, then $X \subseteq X^{\perp\perp}$ and $X^{\perp\perp\perp} = X^\perp$.

Definition 21 (Lists and Cons construct)
If $X \subseteq \mathcal{D}$ and $Y \subseteq \mathcal{D}^*$, then define $X :: Y := \{u :: \vec{v} \mid u \in X, \vec{v} \in Y\}$. ※

Definition 22 (Interpretation of types)
Mappings from type variables to subsets of $\mathcal{D}^*$ are called *valuations*.

Given such a valuation σ, the interpretation of types is defined as follows:

$$\begin{array}{llll} [\alpha]_\sigma & := \sigma(\alpha) & [\mathfrak{A} \cap \mathfrak{B}]_\sigma & := [\mathfrak{A}]_\sigma \cup [\mathfrak{B}]_\sigma \\ [\mathfrak{A}\to\mathfrak{B}]_\sigma & := [\![\mathfrak{A}]\!]_\sigma :: [\mathfrak{B}]_\sigma & [\forall\alpha\mathfrak{A}]_\sigma & := \bigcup_{Y\subseteq\mathcal{D}^*} [\mathfrak{A}]_{\sigma,\alpha\mapsto Y} \end{array}$$

$$[\![\mathfrak{A}]\!]_\sigma := [\mathfrak{A}]_\sigma^\perp$$

The interpretation of typing contexts is defined as follows:

$$[\![\mathfrak{G}]\!]_\sigma := \{\rho \mid \forall (x:\mathfrak{A}) \in \mathfrak{G}, \rho(x) \in [\![\mathfrak{A}]\!]_\sigma\}$$

※

Remark 27 Note that $[\mathfrak{A}\{\alpha := \mathfrak{B}\}]_\sigma = [\mathfrak{A}]_{\sigma,\alpha\mapsto[\mathfrak{B}]_\sigma}$ and $[\![\mathfrak{A}\{\alpha := \mathfrak{B}\}]\!]_\sigma = [\![\mathfrak{A}]\!]_{\sigma,\alpha\mapsto[\mathfrak{B}]_\sigma}$.
Also note that $[\![\mathfrak{A} \cap \mathfrak{B}]\!]_\sigma = [\![\mathfrak{A}]\!]_\sigma \cap [\![\mathfrak{B}]\!]_\sigma$ and $[\![\forall\alpha\mathfrak{A}]\!]_\sigma = \bigcap_{Y\subseteq\mathcal{D}^*} [\![\mathfrak{A}]\!]_{\sigma,\alpha\mapsto Y}$.

An orthogonality model is a sufficiently rich structure for Adequacy to hold:

Lemma 28 (Adequacy Lemma)
If $\mathfrak{G} \vdash_{\mathcal{S}} M : \mathfrak{A}$, then for all valuations σ and for all mappings $\rho \in [\![\mathfrak{G}]\!]_\sigma$ we have $[\![M]\!]_\rho \in [\![\mathfrak{A}]\!]_\sigma$.

Proof: By induction on $\mathfrak{G} \vdash_{\mathcal{S}} M : \mathfrak{A}$, using the axioms (A1),...,(A6) from Definition 19. See Appendix C. □

4.5.2 The special case of applicative structures

In the next section we present instances of orthogonality models. They will have in common that $\mathcal{E}$ is an applicative structure, as we have seen with I-filters. This motivates the following notion:

Definition 23 (Applicative orthogonality model)
An *applicative orthogonality model* is a 4-tuple $(\mathcal{E}, \mathcal{D}, @, [\![\_]\!]_\_)$ where:

- $\mathcal{E}$ is a set, $\mathcal{D}$ is a subset of $\mathcal{E}$, $@$ is a (total) function from $\mathcal{E}\times\mathcal{E}$ to $\mathcal{E}$, and $[\![\_]\!]_\_$ is a function (parametrised by an environment) from λ-terms to the support.
- $(\mathcal{E}, \mathcal{D}, \perp\!\!\!\perp, [\![\_]\!]_\_)$ is an orthogonality model,
where the relation $u \perp\!\!\!\perp \vec{v}$ is defined as $(u@\vec{v}) \in \mathcal{D}$
(writing $u@\vec{v}$ for $(\dots(u@v_1)@\dots@v_n)$ if $\vec{v} = v_1::\dots v_n::[]$).

※

Remark 29 Axioms (A1) and (A2) are ensured provided that $[\![M\ N]\!]_\rho = [\![M]\!]_\rho @ [\![N]\!]_\rho$ and $[\![x]\!]_\rho = \rho(x)$. These conditions can hold by definition (as in term models, cf. the next Section), or can be proved (as in Theorem 21, which also proves axioms (A3)-(A6)).

4.5.3 Instances of orthogonality models

We now give instances of (applicative) orthogonality models with well-chosen sets of values, applications, and interpretations of terms, with the aim of deriving the strong normalisation of a term M of type $\mathfrak{A}$ in $\mathcal{S}$ from the property $[\![M]\!] \in [\![\mathfrak{A}]\!]$.

The first two instances are term models: terms are interpreted as pure λ-terms (see Definition 24), so the support of those term models is the set of all pure λ-terms seen as an applicative structure (using term application: $M_1 @^{\text{TERM}} M_2 := M_1\ M_2$).

Definition 24 (Interpretation of terms in a term model)

$$
\begin{array}{lcl}
[\![x]\!]_\rho^{\text{TERM}} & := & \rho(x) \\
[\![M_1\ M_2]\!]_\rho^{\text{TERM}} & := & [\![M_1]\!]_\rho^{\text{TERM}}\ [\![M_2]\!]_\rho^{\text{TERM}} \\
[\![\lambda x.M]\!]_\rho^{\text{TERM}} & := & \lambda x.[\![M]\!]_{\rho,x\mapsto x}^{\text{TERM}} \\
[\![M[x := N]]\!]_\rho^{\text{TERM}} & := & [\![M]\!]_{\rho,x\mapsto[\![N]\!]_\rho^{\text{TERM}}}^{\text{TERM}} \\
[\![W_x(M)]\!]_\rho^{\text{TERM}} & := & [\![M]\!]_\rho^{\text{TERM}} \\
[\![C_x^{y,z}(M)]\!]_\rho^{\text{TERM}} & := & [\![M]\!]_{\rho,y\mapsto\rho(x),z\mapsto\rho(x)}^{\text{TERM}}
\end{array}
$$

※

Remark 30 $[\![M\{x := N\}]\!]^{\text{TERM}}_{\rho} = [\![M]\!]^{\text{TERM}}_{\rho,x\mapsto[\![N]\!]^{\text{TERM}}_{\rho}}$

In the first instance, values are those pure λ-terms that are strongly normalising (for β). If we concentrate on the interpretation (as themselves) of the pure λ-terms that are typed in $\mathcal{S}$, we have an orthogonality model that rephrases standard proofs of strong normalisation by orthogonality or reducibility candidates [Par97, LM08].

In the second instance, values are those pure λ-terms that can be typed with intersection types, for instance in system $\lambda^{\subseteq}_{\cap}$.

Theorem 31 The structures

- $\mathcal{M}^{\perp\!\!\!\perp}_{\mathsf{SN}} := (\Lambda^{\lambda}, \mathsf{SN}^{\lambda}, @^{\text{TERM}}, [\![\_]\!]^{\text{TERM}}_{\_})$
- $\mathcal{M}^{\perp\!\!\!\perp}_{\cap} := (\Lambda^{\lambda}, \Lambda^{\lambda}_{\cap}, @^{\text{TERM}}, [\![\_]\!]^{\text{TERM}}_{\_})$ (where $\Lambda^{\lambda}_{\cap}$ is the set of pure λ-terms typable in $\lambda^{\subseteq}_{\cap}$)

are applicative orthogonality models.

Indeed, the applicative structures $\mathcal{M}^{\perp\!\!\!\perp}_{\mathsf{SN}}$ and $\mathcal{M}^{\perp\!\!\!\perp}_{\cap}$ already satisfy axioms (A1), (A2), and (A4) to (A6), because of Definition 24. Axiom (A3) holds in $\mathcal{M}^{\perp\!\!\!\perp}_{\mathsf{SN}}$ and $\mathcal{M}^{\perp\!\!\!\perp}_{\cap}$ because of their respective expansion properties:

Lemma 32 (Expansion)

1. If $M\{x := P\}\ \vec{N} \in \mathsf{SN}^{\lambda}$ and $P \in \mathsf{SN}^{\lambda}$, then $(\lambda x.M)\ P\ \vec{N} \in \mathsf{SN}^{\lambda}$.
2. If $M\{x := P\}\ \vec{N} \in \Lambda^{\lambda}_{\cap}$ and $P \in \Lambda^{\lambda}_{\cap}$, then $(\lambda x.M)\ P\ \vec{N} \in \Lambda^{\lambda}_{\cap}$.

Admittedly, once $\lambda^{\subseteq}_{\cap}$ has been proved to characterise SN^{λ} (Theorems 10 and 53), the two points are identical and so are the two models $\mathcal{M}^{\perp\!\!\!\perp}_{\mathsf{SN}}$ and $\mathcal{M}^{\perp\!\!\!\perp}_{\cap}$. But involving the bridge of this characterisation goes much beyond what is needed for either point: point 1 is a known fact of the literature; point 2 is a simple instance of Subject Expansion (Theorem 51 in the next section) not requiring Subject Reduction (Theorem 9) while both are involved at some point in the more advanced property $\mathsf{SN}^{\lambda} = \Lambda^{\lambda}_{\cap}$. In brief, as we are interested in comparing proof *techniques* for the strong normalisation of System $\mathcal{S}$, the question of which properties are used and in which order matters.

Now using the Adequacy Lemma (Lemma 28), we finally get:

Corollary 33 If $\mathfrak{G} \vdash_{\mathcal{S}} M : \mathfrak{A}$, then:

$\mathcal{M}^{\perp\!\!\!\perp}_{\mathsf{SN}}$ For all valuations σ and all mappings $\rho \in [\![\mathfrak{G}]\!]_{\sigma}$ we have $[\![M]\!]^{\text{TERM}}_{\rho} \in \mathsf{SN}^{\lambda}$.

$\mathcal{M}^{\perp\!\!\!\perp}_{\cap}$ For all valuations σ and all mappings $\rho \in [\![\mathfrak{G}]\!]_{\sigma}$ there exist Γ and A such that $\Gamma \vdash_{\cap\subseteq} [\![M]\!]^{\text{TERM}}_{\rho} : A$, and therefore $[\![M]\!]^{\text{TERM}}_{\rho} \in \mathsf{SN}^{\lambda}$.

For $\mathcal{M}^{\perp\!\!\!\perp}_{\cap}$ we conclude of course by using Theorem 10.

Now notice that, if M is a pure λ-term, this entails $M \in \mathsf{SN}^{\lambda}$ by choosing, in both models, σ to map every type variable to the empty set, and ρ to map every term variable to itself. Indeed we have:

Remark 34 In both structures $\mathcal{M}^{\perp\!\!\!\perp}_{\mathsf{SN}}$ and $\mathcal{M}^{\perp\!\!\!\perp}_{\cap}$ we can check that:
For all lists $\vec{N}$ of values, and any term variable x, $x \perp\!\!\!\perp \vec{N}$.
Hence, for all valuations σ and all types $\mathfrak{A}$, $x \in [\![\mathfrak{A}]\!]_{\sigma}$.

However if M is not a pure λ-term, it is not obvious to derive an interesting normalisation result for M, given that the explicit substitutions / weakenings / contractions in M have disappeared in $[\![M]\!]^{\text{TERM}}_{\rho}$ (and in the case of $\mathcal{M}^{\perp\!\!\!\perp}_{\mathsf{SN}}$, relating SN^{λ} to $\mathsf{SN}^{\lambda S}$ or $\mathsf{SN}^{\lambda lxr}$ is another task to do).

An idea would be to tweak the interpretation of terms so that every term is interpreted as itself, even if it has explicit substitutions / weakenings / contractions:

$$\begin{array}{lcl} [\![M[x:=N]]\!]^{\text{TERM}}_{\rho} & := & [\![M]\!]^{\text{TERM}}_{\rho,x\mapsto x}[x:=[\![N]\!]^{\text{TERM}}_{\rho}] \\ [\![W_x(M)]\!]^{\text{TERM}}_{\rho} & := & W_x([\![M]\!]^{\text{TERM}}_{\rho}) \\ [\![C^{y,z}_x(M)]\!]^{\text{TERM}}_{\rho} & := & C^{y,z}_x([\![M]\!]^{\text{TERM}}_{\rho,y\mapsto y,z\mapsto z}) \end{array}$$

But proving axioms (A1) to (A6) then becomes much more difficult. This is however the direction taken by [Kes09] for the explicit substitution calculus λ_{ex}, where the methodological cornerstone is a property called **IE**, which is nothing else but axiom (A4) in $\mathcal{M}^{\perp\!\!\!\perp}_{\mathsf{SN}}$. For $\mathcal{M}^{\perp\!\!\!\perp}_{\cap}$, it might be possible to prove the axioms by inspecting typing derivations and/or using Subject Expansion (Theorem 51 in the next section).

A quicker way is to depart from term models and turn the filter model $\mathcal{M}^{\mathsf{i}}_{\mathcal{F}}$ from Section 4.4 into an orthogonality model: a term is interpreted as the filter of the intersection types that it can be assigned (e.g. in $\lambda^{\subseteq}_{\cap}$, see Definition 14), and orthogonality is defined in terms of filters being non-empty.

Strong Normalisation will then follow, in a very uniform way for the three calculi λ, λS, and λlxr, from the fact that terms typable with intersection types are themselves strongly normalising (Theorems 10, 13, 16 for $\lambda^{\subseteq}_{\cap}$).

Theorem 35 The structure $\mathcal{M}^{\perp\!\!\!\perp}_{\mathcal{F}} := (\mathcal{E}, \mathcal{D}, @, [\![\_]\!]_{\_})$ (with the four components as defined in Section 4.1) is an applicative orthogonality model.

Proof: Indeed, $\mathcal{M}^{\perp\!\!\!\perp}_{\mathcal{F}}$ satisfies axioms (A1) to (A6) as immediate consequences of Theorem 21. □

Remark 36 For $\mathcal{M}^{\perp\!\!\!\perp}_{\mathcal{F}}$ we now have: For all list of values $\vec{v}$, $\top \perp\!\!\!\perp \vec{v}$.
Hence, for all valuations σ and all types $\mathfrak{A}$, $\top \in [\![\mathfrak{A}]\!]_{\sigma}$.

Now using the Adequacy Lemma (Lemma 28), we finally get:

Corollary 37 If $\mathfrak{G} \vdash_{\mathcal{S}} M : \mathfrak{A}$, then:
For all valuations σ and all mappings $\rho \in [\![\mathfrak{G}]\!]_{\sigma}$ we have $[\![M]\!]_{\rho} \neq \bot$.
Hence, there exist Γ and A such that $\Gamma \vdash_{\cap\subseteq} M : A$.
Finally, $M \in \mathsf{SN}^{\lambda}$ (resp. $M \in \mathsf{SN}^{\lambda S}$, $M \in \mathsf{SN}^{\lambda lxr}$, according to the calculus considered).

Proof: The first statement holds because $\bot \notin \mathcal{D}$ and $[\![\mathfrak{A}]\!]_{\sigma} \subseteq \mathcal{D}$. To prove the second, we need to show that there exist such a σ and such a ρ; take σ to map every type variable to the empty set and take ρ to map every term variable to $\top$. The final result comes from Theorem 10 (resp. Theorem 13, Theorem 16). □

5 Completeness

In Section 3 we have shown that for the three calculi, if a term is typable with intersection types, then it is strongly normalising. We have briefly mentioned that the converse is true. In this section we give a proof for the three calculi. Moreover, the typing trees obtained by these completeness theorems satisfy some interesting properties that will be used in the next sections (for the complexity results): they are *optimal* and *principal*.

The proof of completeness for λS is simpler than the one for the pure λ-calculus. This is why, in this section, that we will treat the λS calculus first.

5.1 Two properties of typing trees: Optimality and Principality

In the next sub-section we will notice that the typing trees produced by the proof of completeness all satisfy a particular properties. In this section we define these properties. The first of these is *optimality*.

This property involves the following notions:

Definition 25 (Subsumption and forgotten types)

- If π is a typing tree, we say that π *does not use subsumption* if it features an occurrence of the abstraction rule where the condition $A \subseteq U$ is either $A \approx U$ or $A \subseteq \omega$.
- We say that a type A is *forgotten* in an instance of rule (Abs) or rule (Subst) if in the side-condition of the rule we have $U = \omega$.
- If a typing tree π uses no subsumption, we collect the list of its forgotten types, written forg(π), by a standard prefix and depth-first search of the typing tree π.

※

The optimal property also involves refining the grammar of types:

Definition 26 (Refined intersection types) A^+, A^-, A^{--} and U^{--} are defined by the following grammar:

$$\begin{array}{lll} A^+, B^+ & ::= & \tau \mid A^{--} \to B^+ \\ A^{--}, B^{--} & ::= & A^- \mid A^{--} \cap B^{--} \\ A^-, B^- & ::= & \tau \mid A^+ \to B^- \\ U^{--}, V^{--} & ::= & A^{--} \mid \omega \end{array}$$

We say that Γ is of the form Γ^{--} if for all x, $\Gamma(x)$ is of the form U^{--}. The *degree* of a type of the form A^+ is the number of arrows in negative positions:

$\delta^+(\tau)$	$:=$	0
$\delta^+(A^{--} \to B^+)$	$:=$	$\delta^-(A^{--}) + \delta^+(B^+) + 1$
$\delta^-(A^{--} \cap B^{--})$	$:=$	$\delta^-(A^{--}) + \delta^-(B^{--})$
$\delta^-(\tau)$	$:=$	0
$\delta^-(A^+ \to B^-)$	$:=$	$\delta^+(A^+) + \delta^-(B^-)$

※

Remark 38 If $A^+ \approx B^+$, then $\delta^+(A) = \delta^+(B)$, and if $A^{--} \approx B^{--}$, then $\delta^-(A^{--}) = \delta^-(B^{--})$.

We can finally define the optimal property:

Definition 27 (Optimal typing) A typing tree π concluding $\Gamma \vdash_{\cap\subseteq} M : A$ is optimal if

- There is no subsumption in π
- A is of the form A^+
- For every $(x : B) \in \Gamma$, B is of the form B^{--}
- For every forgotten type B in π, B is of the form B^+.

We write $\Gamma \vdash_{\text{opt}} M : A^+$ if there exists such π.

The *degree* of such a typing tree is defined as

$$\delta(\pi) = \delta^+(A^+) + \Sigma_{x\,:\,B^{--} \in \Gamma} \delta^-(B^{--}) + \Sigma_{C^+ \in \text{forg}(\pi)} \delta^+(C^+)$$

※

In this definition, A^+ is an output type, A^- is a basic input type (i.e. for a variable to be used once), and A^{--} is the type of a variable that can be used several times. The intuition behind this asymmetric grammar can be found in linear logic:

Remark 39 Intersection in a typing tree means duplication of resource. So intersections can be compared to exponentials in linear logic [Gir87]. Having an optimal typing tree means that duplications are not needed in certain parts of the optimal typing tree. In the same way, in linear logic, we do not need to have exponentials everywhere: A simple type T can be translated as a type T^* of linear logic as follows:

τ^*	$:= \tau$
$(T \to S)^*$	$:= !T^* \multimap S^*$

We can find a more refined translation; it can also be translated as T^+ and T^- as follow :

τ^+	$:= \tau$	τ^-	$:= \tau$
$(T \to S)^+$	$:= !T^- \multimap S^+$	$(T \to S)^-$	$:= T^+ \multimap S^-$

And we have in linear logic : $T^- \vdash T^*$ and $T^* \vdash T^+$. So the translation T^+ is sound and uses less exponentials that the usual and naive translation. In some way, it is more "optimal". The main drawback is that we cannot compose proofs of optimal translations easily.

We now introduce the second of these properties: the notion of principal typing.

Definition 28 (Principal typing)
A typing tree π of M is *principal*[8] if it is optimal and of minimal degree: For every optimal typing tree π' of M, $\delta(\pi) \leq \delta(\pi')$.

※

5.2 λS

In order to prove the completeness of the typing system with respect to $\mathsf{SN}^{\lambda S}$, we first show that terms in normal form (for some adequate notion of normal form) can be typed, and then we prove Subject Expansion for a notion of reduction that can reduce any term in $\mathsf{SN}^{\lambda S}$ to a normal form (which we know to be typed). In λS, Subject Expansion is true only for $\longrightarrow_{B,S}$ (not for $\longrightarrow_W$). We will prove that it is enough for completeness. The main reason is that $\longrightarrow_W$ can be postponed w.r.t. $\longrightarrow_{B,S}$:

Lemma 40 (Erasure postponement)

- If $M \longrightarrow_W \longrightarrow_B M'$, then $M \longrightarrow_B \longrightarrow_W M'$.
- If $M \longrightarrow_W \longrightarrow_S M'$, then $M \longrightarrow_S^+ \longrightarrow_W^+ M'$.
- If $M \longrightarrow_{S,W}^* M'$, then $M \longrightarrow_S^* \longrightarrow_W^* M'$

[8] In the literature, *principality* is often used (see e.g. [Wel02]) for a typing judgement that characterises, for a given term, all of its derivable typing judgements: they are all obtained from the principal one by instantiation of its type variables. This notion was our guiding intuition (and we therefore kept the terminology) and does relate to a notion of minimality in the size of types (a type is smaller in size than its instances). But for our purpose, we needed a notion that also says something about typing trees rather than the typing judgements that they derive, so optimality of typing trees led to a notion of principality on trees rather than judgements. Saying that all (optimal) typing trees can be obtained by instantiation of the principal one would require quotienting the trees to identify irrelevant differences (integrating AC-equivalence of types and contexts), so it was simpler to express a minimality condition on degrees, both to define the notion and to use it in Section 6 for Complexity results.

Proof:
The first two points are proved by inspection of the rules: a substitution never blocks computation.

The third point: Let L a S, W reduction sequence from M to M'.

If L is not a of the form $\longrightarrow_S^* \longrightarrow_W^*$, then there exists $\longrightarrow_W \longrightarrow_S$ in L and by using the second point we can replace it by $\longrightarrow_S^+ \longrightarrow_W^+$ to obtain a reduction sequence L'. Therefore we have a non-deterministic rewriting of L.

This rewriting increases or does not change the size of L. According to Lemma 11, M is strongly normalising for S, W. Therefore, after a certain number of steps, the size of L does not change. So, after a certain number of steps, the rewriting is just replacing $\longrightarrow_W \longrightarrow_S$ by $\longrightarrow_S \longrightarrow_W$ and this terminates. Hence, this rewriting terminates.

By taking a normal form of this rewriting we have $M \longrightarrow_S^* \longrightarrow_W^* M'$. □

Therefore, the normal forms for $\longrightarrow_{B,S}$ are "normal enough" to be easily typed:

Lemma 41 (Typability of B, S-normal term)
If M cannot be reduced by $\longrightarrow_{B,S}$, then there exist Γ and A such that $\Gamma \vdash_{\mathsf{opt}} M : A$.

Proof: By induction on M.

We use the fact that if M cannot be reduced by $\longrightarrow_{B,S}$, then M is of one of the following form:

- $\lambda x.M_1$
- $x[y_1 := M_1] \cdots [y_n := M_n] N_1 \cdots N_n$

Each of them can easily be typed by a principal typing tree using the induction hypothesis. □

Remark 42 The algorithm given by the previous proof gives us a principal typing tree.

Theorem 43 (Subject Expansion)
If $M \longrightarrow_{B,S} M'$ and $\Gamma' \vdash_{\cap\subseteq} M' : A$, then there exist $\Gamma \approx \Gamma'$ such that $\Gamma \vdash_{\cap\subseteq} M : A$.
Moreover, the optimality property, the degree, and the principality property are all preserved.

Proof: First by induction on $\longrightarrow_{B,S}$ and $\approx$, then by induction on A.

We adapt the proof of Subject Reduction. The optimality property, the degree, and the principality property are preserved in both directions (Subject Expansion and Subject Reduction): indeed, since we are considering $\longrightarrow_{B,S}$ and not $\longrightarrow_W$, the interface (typing context, type of the term and forgotten types) is not changed and we do not add any subsumption. □

Theorem 44 (Completeness) If $M \in \mathsf{SN}^{\lambda S}$, then there exist Γ and A such that $\Gamma \vdash_{\mathsf{opt}} M : A$.

Proof: By induction on the longest reduction sequence of M. If M can be reduced by $\longrightarrow_{B,S}$ we can use the induction hypothesis. Otherwise, M is typable. □

Remark 45 The algorithm given by the previous proof gives us a principal typing tree.

Corollary 46 If $M \longrightarrow_{B,S} M'$ and $M' \in \mathsf{SN}^{\lambda S}$, then $M \in \mathsf{SN}^{\lambda S}$.

5.3 Pure λ-calculus

As in the case of λS, proving completeness of the typing system with respect to SN^{λ} relies on the typability of (some notion of) normal forms and on some property of Subject Expansion.

For the λ-calculus, the normal forms that we consider are simply the β-normal forms. Typing them with the optimal typing trees of System $\lambda_{\cap}^{\subseteq}$ is therefore very reminiscent of [DCHM05] that applies a similar technique to type β-normal forms with the principal types of a system with idempotent intersections.

Definition 29 (Accumulators) The fact that a λ-term M is a λ-free head-normal form with head-variable x (i.e. M is of the form $xM_1 \dots M_n$) is abbreviated as $Acc(M,x)$. Equivalently, the judgement $Acc(M,x)$ can be defined with the following rules:

$$\frac{}{Acc(x,x)} \qquad \frac{Acc(M,x)}{Acc(MN,x)}$$

※

Remark 47 (Shape of a normal term) If M is a β-normal form, then either M is of the form $\lambda x.N$ (for some normal form N) or there exists x such that $Acc(M,x)$ (induction on M -see e.g. [Böh68]).

Lemma 48 (Typability of accumulators)
If $Acc(M,x)$ and π is a derivation of $\Gamma, x{:}U^{--} \vdash_{\cap\subseteq} M{:}F$, then

1. F is of the form F^-;
2. for all G^-, there exists V^{--} and a derivation π' of $\Gamma, x{:}V^{--} \vdash_{\cap\subseteq} M{:}G^-$;
   moreover, $\mathsf{forg}(\pi') = \mathsf{forg}(\pi)$ and if π does not use subsumption, neither does π'.

Proof:

1. By induction on $Acc(M,x)$.
2. By induction on $Acc(M,x)$.
   - For $\dfrac{}{Acc(x,x)}$: Then $\Gamma^{--} = ()$ and $x{:}G^- \vdash_{\cap\subseteq} x{:}G^-$.
   - For $\dfrac{Acc(M,x)}{Acc(MN,x)}$: Then, there exist Γ_1^{--}, Γ_2^{--}, U_1^{--}, U_2^{--} and A such that $\Gamma^{--} = \Gamma_1^{--} \cap \Gamma_2^{--}$, $U^{--} = U_1^{--} \cap U_2^{--}$, $\Gamma_1^{--}, x{:}U_1^{--} \vdash_{\cap\subseteq} M{:}A \to F$ and $\Gamma_2^{--}, x{:}U_2^{--} \vdash_{\cap\subseteq} N{:}A$. By the first point, $A \to F$ is of the form B^-. Therefore, A is of the form A^+. Hence, $A^+ \to G^-$ is of the form C^-. By induction hypothesis, there exist V_1^{--} such that $\Gamma_1^{--}, x{:}V_1^{--} \vdash_{\cap\subseteq} M{:}A^+ \to G^-$. Therefore, $\Gamma, x{:}V_1^{--} \cap U_2^{--} \vdash_{\cap\subseteq} MN{:}G^-$.

□

Lemma 49 (Typability of a normal term)
If M is a normal form, then there exists Γ and F such that $\Gamma \vdash_{\mathsf{opt}} M{:}F$.

Proof: By induction on M. Since M is a normal form, we are in one of the following cases:

- M is of the form $\lambda x.N$; by the induction hypothesis we can type N so we can type $\lambda x.N$;
- There exists x such as $Acc(M,x)$ and
  - either $M = x$, which can trivially be typed;
  - or $M = M_1 M_2$ with $Acc(M_1,x)$, and by the induction hypothesis we can type M_1 and M_2; therefore we can give any type to M_1 so we can type $M_1 M_2$.

□

Lemma 50 (Anti-substitution lemma) If $\Gamma \vdash_{\cap\subseteq} M\{x := N\}{:}A$, then there exist Γ_1, Γ_2 and U such that $\Gamma_1 \vdash_{\cap\subseteq} N{:}U$, $\Gamma_2 \vdash_{\cap\subseteq} M{:}A$ and $\Gamma_1 \cap \Gamma_2 \approx \Gamma$.

Proof: First by induction on M then by induction on A. We adapt the proof of Lemma 8. Notice that if $x \notin fv(M)$ we take $U = \omega$. Therefore, N might not be typable by an A-type. □

As we have seen for λS, proving completeness relies on the Subject Expansion property for a notion of reduction that can reduce any term in SN^λ to a normal form (which we know to be typed).

In the pure λ-calculus, not all β-reductions satisfy Subject Expansion. For example, in the following reduction:

$$(\lambda z.(\lambda y.a)(zz))(\lambda y.yy) \longrightarrow_\beta (\lambda z.a)(\lambda y.yy)$$

the second term is typable, but not the first one (because it is not strongly normalising).

As in λS, it is erasure that breaks the Subject Expansion ($(\lambda x.M)N \longrightarrow M$ with $x \notin fv(M)$).

The problem here is that we cannot just study Subject Expansion for β-reductions that do not erase, because forbidding erasure can block a reduction sequence (for example, $(\lambda x.(\lambda y.y))ab$).

So we have to define a restricted version of β-reduction that satisfies Subject Expansion, but that is still general enough to reach β-normal forms (which can be easily typed).

If M and N are λ-terms and E a finite set of variables then we define the judgements $M \rightsquigarrow_E N$ and $M \Rightarrow_E N$ with the rules of Fig. 7.

$$\frac{x \in fv(M)}{(\lambda x.M)N \rightsquigarrow_\emptyset M\{x := N\}} \qquad \frac{x \notin fv(M) \quad N \text{ is a } \beta\text{-normal form}}{(\lambda x.M)N \rightsquigarrow_{fv(N)} M} \qquad \frac{x \notin fv(M) \quad N \Rightarrow_E N'}{(\lambda x.M)N \rightsquigarrow_E (\lambda x.M)N'}$$

$$\frac{M \rightsquigarrow_E M'}{MN \rightsquigarrow_E M'N} \qquad \frac{N \rightsquigarrow_E N'}{MN \rightsquigarrow_E MN'} \qquad \frac{M \rightsquigarrow_E M' \quad x \notin E}{\lambda x.M \rightsquigarrow_E \lambda x.M'}$$

$$\frac{M \rightsquigarrow_E M'}{M \Rightarrow_E M'} \qquad \frac{M \Rightarrow_E M'}{\lambda x.M \Rightarrow_{E-\{x\}} \lambda x.M'} \qquad \frac{Acc(M,x) \quad N \Rightarrow_E N'}{MN \rightsquigarrow_{E\cup\{x\}} MN'}$$

Figure 7: Restricted β-reduction

These may not seem natural on a syntactic point of view. However, they are quite intuitive if you consider that they satisfy the following lemma:

Theorem 51 (Subject Expansion) If $E = \{x_1, \ldots, x_n\}$ and $\Gamma, x_1 : U_1, \ldots, x_n : U_n \vdash_{\cap\subseteq} M' : A$

- If $M \Rightarrow_E M'$ then there exist B, Γ', V_1, …, V_n such that $\Gamma', x_1 : V_1, \ldots, x_n : V_n \vdash_{\cap\subseteq} M : B$ and $\Gamma \approx \Gamma'$.
- If $M \rightsquigarrow_E M'$ then there exists Γ', V_1, …, V_n, such that $\Gamma', x_1 : V_1, \ldots, x_n : V_n \vdash_{\cap\subseteq} M : A$ and $\Gamma \approx \Gamma'$.

Moreover, if the typing of M' is optimal, then the typing of M' can be required to be optimal.

Proof: First by induction on $M \rightsquigarrow_E M'$ and $M \Rightarrow_E M'$ then by induction on A. We adapt the proof of Subject Reduction (Theorem 9). □

Lemma 52 (Safe execution of a term) If M can be reduced by $\longrightarrow_\beta$ then there exist M' and E such that $M \Rightarrow_E M'$ and if M is not of the form $\lambda x.M$ then $M \rightsquigarrow_E M'$.

Proof: By induction on M.

- M cannot be a variable.
- If M is of the form $\lambda x.N$. Then N reduced by $\longrightarrow_\beta$. By induction hypothesis there exist N' and E such that $N \Rightarrow_E N'$. Therefore $\lambda x.N \Rightarrow_{E-\{x\}} \lambda x.N'$.
- If M is of the form $M_1 M_2$. We are in one of the following cases:
  - M_1 is of the form $\lambda x.M_3$. Then we have $(\lambda x.M_3)M_2 \rightsquigarrow_E M_3\{x := M_2\}$ with $E = \emptyset$ or $E = fv(M_2)$.

– M_1 is not of the form $\lambda x.M_3$ and M_1 reduced by $\longrightarrow_\beta$. By induction hypothesis, there exist M_1' and E such that $M_1 \rightsquigarrow_E M_1'$. Therefore $M_1M_2 \rightsquigarrow_E M_1'M_2$.
– M_1 is not of the form $\lambda x.M_3$ and M_1 is a β-normal form. Therefore there exists x such that $Acc(M_1, x)$ and M_2 reduced by $\longrightarrow_\beta$. By induction hypothesis, there exist M_2' and E such that $M_2 \Rightarrow_E M_2'$. Hence $M_1M_2 \rightsquigarrow_{E\cup\{x\}} M_1M_2'$.

□

Theorem 53 (Completeness) If $M \in \mathsf{SN}^\lambda$ then there exists Γ and A such that $\Gamma \vdash_{\mathsf{opt}} M : A$.

Proof: By induction on the longest reduction sequence of M. □

We can notice that to prove the completeness of the pure λ-calculus we only need a fragment of $\rightsquigarrow$ and $\Rightarrow$ but by dealing with all $\rightsquigarrow$ and $\Rightarrow$ we have the following result without extra difficulties:

Corollary 54 If $M \Rightarrow_E M'$ and $M' \in \mathsf{SN}^\lambda$ then $M \in \mathsf{SN}^\lambda$.

This result is purely syntactic. However, it is very hard to prove without intersection types (if we consider all $\rightsquigarrow$ and $\Rightarrow$ and not just the head reduction fragment).

5.4 λlxr

In this section we provide the guidelines to obtain a similar completeness theorem for λlxr, leaving the details for further work. The methodology is similar to the cases of the λ-calculus and λS: we identify a notion of reduction for which Subject Expansion holds, and whose notion of normal forms can be easily typed.

As in the λ-calculus, some of the rules do not satisfy Subject Expansion:

$$\begin{array}{lcl} W_x(M)[x := N] & \longrightarrow & W_{fv(N)}(M) \\ MW_x(N) & \longrightarrow & W_x(MN) \\ M[x := W_y(N)] & \longrightarrow & W_y(M[x := N]) \\ C_x^{y,z}(W_y(M)) & \longrightarrow & M\{z := x\} \end{array}$$

Subject Expansion for the other reduction rules should be straightforward.

The fact that the last 3 rules above do not satisfy Subject Expansion is not problematic for the completeness theorem: like rule W in λS, we should prove that they can be postponed after the other rules, and that removing them from the system defines a new notion of normal forms that can still be typed.

On the other hand, the first rule above is more problematic: if we forbid it, the reduction can be blocked (like forbidding erasure can block a reduction in the pure λ-calculus). So a Subject Expansion result without that rule is not enough to prove the completeness of λlxr. Hence, we have two possibilities to achieve that goal:

- We adapt the proof of the pure λ-calculus. We have to define $\rightsquigarrow_E$ and $\Rightarrow_E$ in λlxr.
- We adapt the proof of λS. We cannot do this in the usual λlxr: if we forbid erasure, reduction can be blocked. So we have to add the rules that move $W_x(M)[x := N]$ (like $M[x := N]$ with $x \notin fv(M)$ in λS). These added rules respect Subject Reduction, so we still have soundness.

Both approaches would provide Completeness (strong normalisation implies typability). Moreover, as in the pure λ-calculus and λS, the proof would provide an algorithm that constructs an optimal typing tree from a strongly normlising term in λlxr.

6 Complexity results

With the Subject Reduction theorems of the different calculi, we have proved that for every β- (or B-) reduction, the measure of the typing tree strictly decreases. Hence, more than strong normalisation, this gives us a bound on the number of β- (or B-) reductions in a reduction sequence. So we have a complexity result which is an inequality. We would like to refine this result and have

an equality instead. The main idea is to only perform β- (or B-) reductions that decreases the measure of the typing tree by exactly one. Given a term M and any typing tree for it, it is not always possible to find such a reduction step. But it is always possible provided the typing tree is optimal. Fortunately, every term M that is typable is typable with an optimal typing tree: with soundness we can prove that M is strongly normalising, and then, with completeness, we can prove that M is typable with an optimal typing tree. This is the main reason for introducing the notion of optimality. As in Section 5, the case of λS is simpler than the case of pure λ-calculus, so we will deal with it first.

6.1 λS

In λS, we take advantage of the fact that $\longrightarrow_W$ can be postponed w.r.t. to $\longrightarrow_{B,S}$ steps. This allows us to concentrate on $\longrightarrow_{B,S}$ and the normal forms for it.

Lemma 55 (Refined Subject Reduction) If $\Gamma \vdash^n_{\mathsf{opt}} M : A$ then:

- If $M \longrightarrow_B M'$, then there exist Γ' and m such that $\Gamma \approx \Gamma'$, $m < n$ and $\Gamma' \vdash^m_{\mathsf{opt}} M' : A$
- If $M \longrightarrow_S M'$, then there exists Γ' such that $\Gamma \approx \Gamma'$ and $\Gamma' \vdash^n_{\mathsf{opt}} M' : A$

Moreover the degree of the typing tree does not change, and the principality property is preserved.

Proof: We simply check that, in the proof of of Subject Reduction (Theorem 12), the optimality property, the degree and the principality property are preserved, as already mentioned in the proof of Subject Expansion (Theorem 43). □

Lemma 56 (Most inefficient reduction) Assume $\Gamma \vdash^n_{\mathsf{opt}} M : A$. If M can be reduced by $\longrightarrow_B$ and not by $\longrightarrow_S$, then there exist M' and Γ' such that $\Gamma \approx \Gamma'$, $M \longrightarrow_B M'$ and $\Gamma' \vdash^{n-1}_{\mathsf{opt}} M' : A$.

Proof: Again, we adapt the proof of Subject Reduction (Theorem 12). More precisely, see appendix C. □

Lemma 57 (Resources of a normal term) If $\Gamma \vdash^n_{\mathsf{opt}} M : A$, and M cannot be reduced by $\longrightarrow_{B,S}$, then n is the number of applications in M.

Moreover, if the typing tree is principal, then n is the degree of the typing tree.

Proof: Straightforward. □

Theorem 58 (Complexity result) If $\Gamma \vdash^n_{\mathsf{opt}} M : A$, then $n = n_1 + n_2$ where

- n_1 is the maximum number of $\longrightarrow_B$ in a B,S-reduction sequence from M
- n_2 is the number of applications in the B,S normal form of M.

Moreover, if the typing tree is principal, then n_2 is the degree of the typing tree.

Proof: The previous lemmas give us a B,S-reduction sequence with $n - n_2$ B-steps, from M to the normal form of M. In this reduction sequence every reduction B decreases the measure of the typing tree by exactly one.

Assume we have a B,S-reduction sequence, from M to the normal form of M (λS is confluent), with m B-steps. By Subject Reduction (Theorem 12), the measure of the derivation typing the normal form of M is smaller than $n - m$, but is also n_2. Hence $m \leq n - n_2$.

Assume we have a B,S-reduction sequence, from M to any term M_1. It can be completed into a B,S-reduction sequence with more B-steps, from M to the normal form of M_1. □

6.2 λlxr

In this section we suppose that we have reductions to move $W_x(M)[x := N]$.

Therefore, it is reasonable to consider $\longrightarrow_{rev}$ which is all reductions except the 5 ones that cause a problem for subject expansion.

Hence we can adapt the proofs of λS and obtain the following theorem:

Theorem 59 (Complexity result) If $\Gamma \vdash^n_{\mathsf{opt}} M : A$, then $n = n_1 + n_2$ with n_1 the maximum number of B in a $\longrightarrow_{rev}$ sequence, and n_2 the number of applications in the $\longrightarrow_{rev}$ normal form.

Moreover, if the typing tree is principal, then n_2 is the degree of the typing tree.

6.3 Pure λ-calculus

The case of λ-calculus is harder because we cannot ignore erasure (β-reductions that erase sub-terms cannot always be postponed). Therefore:

- We will need to use the results we have for λS.
- We will have to use degrees and principal typing trees. This is why we defined those two notions in the first place.

We produce and measure the longest β-reduction sequences by simply using the perpetual strategy from [vRSSX99], shown in Fig. 8.

$$\frac{x \in fv(M) \text{ or } M' \text{ is a } \beta\text{-normal form}}{(\lambda x.M)\ M'\ \overrightarrow{M_j} \Rightarrow_{head} M\{x := M'\}\ \overrightarrow{M_j}} \qquad \frac{M' \Rightarrow_{head} M'' \qquad x \notin fv(M)}{(\lambda x.M)\ M'\ \overrightarrow{M_j} \Rightarrow_{head} (\lambda x.M)\ M''\ \overrightarrow{M_j}}$$

$$\frac{M \Rightarrow_{head} M'}{x\ \overrightarrow{M_j}\ M\ \overrightarrow{N_j} \Rightarrow_{head} x\ \overrightarrow{M_j}\ M'\ \overrightarrow{N_j}} \qquad \frac{M \Rightarrow_{head} M'}{\lambda x.M \Rightarrow_{head} \lambda x.M'}$$

Figure 8: A perpetual reduction strategy for λ

Remark 60 Notice that this restricted reduction relation is a fragment of that defined in Fig. 7:

$$\Rightarrow_{head} \subseteq \Rightarrow_{\emptyset} \subseteq \longrightarrow_{\beta}$$

Moreover, if M is not a β-normal form, then there is a λ-term M' such that $M \Rightarrow_{head} M'$.

Although we do not need it here, it is worth mentioning that $\Rightarrow_{head}$ defines a perpetual strategy w.r.t. β-reduction, i.e. if M is not β-strongly normalising and $M \Rightarrow_{head} M'$, then neither is M' [vRSSX99]. In that sense it can be seen as the worst strategy (the least efficient). We show here that it is the worst in a stronger sense: it maximises the lengths of reduction sequences.

Lemma 61 (Resources of a normal term)

If $\Gamma \vdash^n_{\mathsf{opt}} M : A$ with a principal typing tree of degree d, and M cannot be β-reduced, then $n = d$.

Proof: If M is a normal form for β, then M is also a normal form for B, S, so we can apply Lemma 57. □

Lemma 62 (Most inefficient reduction)

If $\Gamma \vdash^n_{\mathsf{opt}} M : A$ a principal typing tree with a degree d_1 and $M \Rightarrow_{head} M'$, then there exist m, Γ, A' and d_2 such that $\Gamma' \vdash^m_{\mathsf{opt}} M' : A'$ a principal typing tree with a degree d_2 and $n - d_1 = m - d_2 + 1$.

Proof: By induction on $M \Rightarrow_{head} M'$. The proof is an adaptation of Lemma 56. But, in Lemma 56 we had $d_1 = d_2$, because there was no erasure. To control how the degree changes when there is erasure, we use Lemma 61. We can use it because $\Rightarrow_{head}$ only erases normal terms. □

Lemma 63 (Relating λ and λS) If $M \longrightarrow_\beta^n M'$, then $M(\longrightarrow_B \longrightarrow_S^*)^n \longrightarrow_W^* M'$.

Proof: We proceed as follows:

- Given two pure λ-terms M and N, note that $M\{x := N\}$ is a pure λ-term, and it is easy to show, by induction on M and using erasure postponement (Lemma 40), that $M[x := N] \longrightarrow_S^* \longrightarrow_W^* M\{x := N\}$.
- Then we show, again by induction on M, that if $M \longrightarrow_\beta M'$, then $M \longrightarrow_B \longrightarrow_S^* \longrightarrow_W^* M'$.

The result is a direct corollary, obtained by induction on n and using Lemma 40. □

Theorem 64 (Complexity result) If $\Gamma \vdash_{\mathsf{opt}}^n M : A$ with a principal typing tree of degree d, then the length of the longest β-reduction sequence from M is $n - d$.

Proof: Two previous lemmas give us a β-reduction sequence of size $n - d$. Let L be another β-reduction sequence from M of size m. So there exists M_1 such that $M \longrightarrow_\beta^m M_1$. By the previous lemma, there exists M_2 such that $M(\longrightarrow_B \longrightarrow_S^*)^m M_2$ and $M_2 \longrightarrow_W^* M_1$. From the complexity result for λS we have $m \leq n - d$. □

Contrary to λS, we cannot have a complexity result on the weaker assumption of optimality, relating the measure to the number of applications in the normal form: This was possible in λS because we considered normal forms for a system that never erases, while here we cannot forbid β-reduction to erase terms.

7 Other measures of complexity

In the pure λ-calculus we measure the (maximal) number of β-steps. The equivalent result for λS and λlxr naturally counts the number of $\longrightarrow_B$ -steps if we do not change the measure on the typing trees. But there are many other reduction rules for λS and λlxr, for which we may want similar complexity results. For some of these rules we can obtain such results without changing the typing system, by changing what we count in the typing trees.

7.1 Number of replacements

To get the number of replacements we measure in the typing tree the number of use of the variable rule.

Theorem 65 (Complexity result on the number of replacements) The longest reduction sequence from M by measuring the number of $\longrightarrow_B$ is the longest reduction sequence from M by measuring $\longrightarrow_{SR}$ (head reduction strategy).

And the number of use of $\longrightarrow_{SR}$ in this sequence plus the number of variables in the normal form (without weakening) is equal to the number of use of the variable rule in an optimal typing tree.

7.2 Number of duplications

By measuring the number of use of the Intersection rule in the typing tree we get a bound on the number of duplications (the number of use of rules that duplicate a term).

However, contrary to the other measures, we cannot have an equality result.

Here is a counter example :

$$(\lambda x.xx)(\lambda y.ayy)$$

If we reduce this term to its normal form we have two duplications. However, if we type this term, we have at least 3 uses of the intersection rule.

7.3 The other measures

- If we measure the number of uses of the Abstraction rule we get a result on the maximum number of $\longrightarrow_B$ in a reduction sequence again. We just have to change the definition of degree of a principal typing tree.
- The explicit substitution rule can be produced or destroyed by the subject reduction. So we cannot use it to get a complexity result.

8 Conclusion

We have defined a typing system with non-idempotent intersection types. We have shown that it characterises strongly normalising terms, in the pure λ-calculus as well as in the explicit substitution calculi λS and λlxr. This characterisation has been achieved in each case by strong versions of Subject Reduction and Subject Expansion, enriched with quantitative information:

- By identifying a measure on typing derivations that is decreased by Subject Reduction, we have obtained a simple proof of strong normalisation that also provides upper bounds on longest reduction sequences.
- By either proving postement of erasures (λS) or identifying appropriate sub-reduction relations (λ), we have shown how Subject Expansion garantees the existence of typing derivations satisfying extra properties (optimality and principality), where the bounds are refined into an exact measure of longest reduction sequences.

In the case of λ-calculus, obtaining this exact equality departs from the issues addressed in e.g. [KW99, NM04] whose technology is similar to ours (as we found out *a posteriori*). Indeed, one of the concerns of this line of research is how the process of type inference compares to that of normalisation, in terms of complexity *classes* (these two problems being parametrised by the size of terms and a notion of *rank* for types). Here we have shown how such a technology can actually provide an exact equality specific to each λ-term and its typing tree. Of course this only emphasises the fact that type inference is as hard as normalisation, but type inference as a process is not a concern of this paper.

Moreover, we have extended those results to λS and λlxr, and the technology can be adapted to other calculi featuring e.g. combinators, or algebraic constructors and destructors (to handle integers, products, sums,...).

We have seen how the use of non-idempotent intersection types simplifies the methodology from [CS07] by cutting a second use of reducibility techniques to prove strong normalisation properties of standard systems (here illustrated by the examples of simple types, System F, and idempotent intersections). We extended the methodology to prove strong normalisation results for λS and λlxr, providing the first direct proofs that we are aware of.

We have seen how the corresponding filter model construction can be done by orthogonality techniques; for this we have defined an abstract notion of orthogonality model which we have not seen formalised in the literature. As illustrated in Section 4.5.3, this notion allows a lot of work (e.g. proving the Adequacy Lemma) to be factorised, while building models like $\mathcal{M}^{\perp\!\!\!\perp}_{\mathsf{SN}}$, $\mathcal{M}^{\perp\!\!\!\perp}_{\cap}$ and $\mathcal{M}^{\perp\!\!\!\perp}_{\mathcal{F}}$. Comparing such instances of orthogonality models, we have seen the superiority of $\mathcal{M}^{\perp\!\!\!\perp}_{\mathcal{F}}$ for proving the strong normalisation results of λS and λlxr. Note that, while $\mathcal{M}^{\perp\!\!\!\perp}_{\mathcal{F}}$ and $\mathcal{M}^{\mathsf{i}}_{\mathcal{F}}$ share the same ingredients $\mathcal{E}$, $\mathcal{D}$, @ and $[\![\_]\!]\_$, they are different in the way types are interpreted; see the discussion in Appendix A.

In [BL11b] we also compared the models in the way they enlighten the transformation of infinite polymorphism into finite polymorphism. We leave this aspect for another paper, as more examples should be computed to illustrate (and better understand) the theoretical result; in particular we need to understand how and why the transformation of polymorphism does not require to reduce terms to their normal forms. An objective could be to identify (and eliminate), in the interpretation of a type from System F, those filters that are not the interpretation of any term of that type.

What could help this, is to force filters to be stable under type instantiation, in the view that interpretations of terms are generated by a single F-type, i.e. a *principal* type.

Another aspect of this future work is to use the filter models to try to lift the complexity results that we have in the target system back into the source system, and see to what extent the quantitative information can be already read in the typing trees of the source system. One hope would be to recover for instance results that are known for the simply-typed calculus [Sch82, Bec01], but with our methodology that can be adapted to other source systems such as System F.

Finally, the appropriate sub-reduction relation for the λ-calculus, which we have used to prove Subject Expansion as generally as possible, also helps understanding how and when the semantics $[\![\_]\!]_{\_}$ of terms is preserved, see Appendix B. This is similar to [ABDC06], and future work should adapt their methodology to accommodate our non-idempotent intersections.

Acknowledgement. The authors are grateful to the anonymous referees for their numerous constructive remarks (and for pointing out references).

References

[ABDC06] F. Alessi, F. Barbanera, and M. Dezani-Ciancaglini. Intersection types and lambda models. *Theoret. Comput. Sci.*, 355(2):108–126, 2006. 2, 3, 32

[Abr93] S. Abramsky. Computational interpretations of linear logic. *Theoret. Comput. Sci.*, 111:3–57, 1993. 3

[AC98] R. Amadio and P.-L. Curien. *Domains and lambda-calculi.* Cambridge University Press, 1998. 2

[Bai02] P. Baillot. Checking polynomial time complexity with types. In R. A. Baeza-Yates, U. Montanari, and N. Santoro, editors, *IFIP TCS*, volume 223 of *IFIP Conference Proceedings*, pages 370–382. Kluwer, 2002. 3

[Bar84] H. P. Barendregt. *The Lambda-Calculus, its syntax and semantics.* Studies in Logic and the Foundation of Mathematics. Elsevier, 1984. Second edition. 5, 9

[BBdH93] N. Benton, G. Bierman, V. de Paiva, and M. Hyland. A term calculus for intuitionistic linear logic. In J. F. G. Groote and M. Bezem, editors, *Proc. of the 1st Int. Conf. on Typed Lambda Calculus and Applications*, volume 664 of *LNCS*, pages 75–90. Springer-Verlag, 1993. 3

[BBLRD96] Z. Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation. *J. Funct. Programming*, 6(5):699–722, 1996. 4

[BCDC83] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *J. of Symbolic Logic*, 48(4):931–940, 1983. 2, 3

[BCL99] G. Boudol, P.-L. Curien, and C. Lavatelli. A semantics for lambda calculi with resources. *Math. Structures in Comput. Sci.*, 9(4):437–482, 1999. 3

[Bec01] A. Beckmann. Exact bounds for lengths of reductions in typed lambda-calculus. *J. of Symbolic Logic*, 66(3):1277–1285, 2001. 3, 32

[BEM10] A. Bucciarelli, T. Ehrhard, and G. Manzonetto. Categorical models for simply typed resource calculi. *ENTCS*, 265:213–230, 2010. 3

[BET10] R. Blute, T. Ehrhard, and C. Tasson. A convenient differential category. *CoRR*, abs/1006.3140, 2010. 3

[BL96] G. Boudol and C. Lavatelli. Full abstraction for lambda calculus with resources and convergence testing. In H. Kirchner, editor, *Trees in Algebra and Programming, 21st Int. Colloquium (CAAP'96)*, volume 1059 of *LNCS*, pages 302–316. Springer-Verlag, 1996. 3

[BL11a] A. Bernadet and S. Lengrand. Complexity of strongly normalising λ-terms via non-idempotent intersection types. In M. Hofmann, editor, *Proc. of the 14th Int. Conf. on Foundations of Software Science and Computation Structures (FOSSACS'11)*, volume 6604 of *LNCS*. Springer-Verlag, 2011. 2, 3, 7, 8, 13

[BL11b] A. Bernadet and S. Lengrand. Filter models: non-idempotent intersection types, orthogonality and polymorphism. In M. Bezem, editor, *Proc. of the 20th Annual Conf. of the European Association for Computer Science Logic (CSL'11)*, LIPIcs. Schloss Dagstuhl LCI, 2011. 2, 3, 4, 31

[BM03] P. Baillot and V. Mogbil. Soft lambda-calculus: a language for polynomial time computation. *CoRR*, cs.LO/0312015, 2003. 3

[Böh68] C. Böhm. Alcune proprietà delle forme β-η-normali nel λK-calcolo. Technical report, IAC, Roma, 1968. 25

[BR95] R. Bloo and K. H. Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In J. van Vliet, editor, *Computing Science in the Netherlands (CSN '95)*, pages 62–72, 1995. 4, 10

[CD78] M. Coppo and M. Dezani-Ciancaglini. A new type assignment for lambda-terms. *Arch. Math. Log.*, 19:139–156, 1978. 2

[CDS79] M. Coppo, M. Dezani, and P. Sallé. Functional characterization of some semantic equalities inside λ-calculus. In H. A. Maurer, editor, *Proc. of the 6th Intern. Col. on Automata, Languages and Programming (ICALP)*, volume 71 of *LNCS*, pages 133–146. Springer-Verlag, 1979. 2

[CS07] T. Coquand and A. Spiwack. A proof of strong normalisation using domain theory. *Logic. Methods Comput. Science*, 3(4), 2007. 2, 3, 4, 13, 14, 15, 17, 31

[dC05] D. de Carvalho. Intersection types for light affine lambda calculus. *ENTCS*, 136:133–152, 2005. 3

[dC09] D. de Carvalho. Execution time of lambda-terms via denotational semantics and intersection types. *CoRR*, abs/0905.4251, 2009. 3

[DCGL04] M. Dezani-Ciancaglini, S. Ghilezan, and S. Likavec. Behavioural Inverse Limit Models. *Theoret. Comput. Sci.*, 316(1–3):49–74, 2004. 2

[DCHM05] M. Dezani-Ciancaglini, F. Honsell, and Y. Motohama. Compositional characterisations of *lambda*-terms using intersection types. *Theoret. Comput. Sci.*, 340(3):459–495, 2005. 2, 24

[DK00] V. Danos and J.-L. Krivine. Disjunctive tautologies as synchronisation schemes. In P. Clote and H. Schwichtenberg, editors, *Proc. of the 9th Annual Conf. of the European Association for Computer Science Logic (CSL'00)*, volume 1862 of *LNCS*, pages 292–301. Springer-Verlag, 2000. 4, 17

[DL03] D. Dougherty and P. Lescanne. Reductions, intersection types, and explicit substitutions. *Math. Structures in Comput. Sci.*, 13(1):55–85, 2003. 4

[ER03] T. Ehrhard and L. Regnier. The differential lambda-calculus. *Theoret. Comput. Sci.*, 309(1-3):1–41, 2003. 3

[Gal98] J. Gallier. Typing untyped lambda terms, or reducibility strikes again. *Ann. Pure Appl. Logic*, 91:231–270, 1998. 2

[Ghi96] S. Ghilezan. Strong normalization and typability with intersection types. *Notre Dame J. Formal Loigc*, 37(1):44–52, 1996. 2

[GILL11] S. Ghilezan, J. Ivetic, P. Lescanne, and S. Likavec. Intersection types for the resource control lambda calculi. In A. Cerone and P. Pihlajasaari, editors, *Proc. of the 8th Int. Colloquium on Theoretical Aspects of Computing (ICTAC'11)*, volume 6916 of *LNCS*, pages 116–134. Springer-Verlag, 2011. 8

[Gir72] J.-Y. Girard. *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur*. Thèse d'état, Université Paris 7, 1972. 3, 4, 13

[Gir87] J.-Y. Girard. Linear logic. *Theoret. Comput. Sci.*, 50(1):1–101, 1987. 3, 4, 17, 23

[GR07] M. Gaboardi and S. R. D. Rocca. A soft type assignment system for *lambda* -calculus. In *Proc. of the 16th Annual Conf. of the European Association for Computer Science Logic (CSL'07)*, volume 4646 of *LNCS*, pages 253–267. Springer-Verlag, 2007. 3

[How80] W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, pages 479–490. Academic Press, 1980. Reprint of a manuscript written 1969. 3

[Kes07] D. Kesner. The theory of calculi with explicit substitutions revisited. In *Proc. of the 16th Annual Conf. of the European Association for Computer Science Logic (CSL'07)*, volume 4646 of *LNCS*, pages 238–252. Springer-Verlag, 2007. 2, 3, 10, 11

[Kes09] D. Kesner. A theory of explicit substitutions with safe and full composition. *Logic. Methods Comput. Science*, 5(3), 2009. 4, 21

[KL05] D. Kesner and S. Lengrand. Extending the explicit substitution paradigm. In J. Giesl, editor, *Proc. of the 16th Int. Conf. on Rewriting Techniques and Applications(RTA'05)*, volume 3467 of *LNCS*, pages 407–422. Springer-Verlag, 2005. 2, 3

[KL07] D. Kesner and S. Lengrand. Resource operators for the λ-calculus. *Inform. and Comput.*, 205:419–473, 2007. 2, 3, 6, 13, 16

[KR11] D. Kesner and F. Renaud. A prismoid framework for languages with resources. *Theoret. Comput. Sci.*, 412(37):4867–4892, 2011. 39

[Kri90] J.-L. Krivine. *Lambda-calcul Types et modèles*. Masson, 1990. 2

[Kri01] J.-L. Krivine. Typed lambda-calculus in classical Zermelo-Frænkel set theory. *Arch. Math. Log.*, 40(3):189–205, 2001. 4, 17

[KW99] A. J. Kfoury and J. B. Wells. Principality and decidable type inference for finite-rank intersection types. In *Proc. of the 26th Annual ACM Symp. on Principles of Programming Languages (POPL'99)*, pages 161–174. ACM Press, 1999. 2, 31

[Laf04] Y. Lafont. Soft linear logic and polynomial time. *Theoret. Comput. Sci.*, 318(1-2):163–180, 2004. 3

[Lei86] D. Leivant. Typing and computational properties of lambda expressions. *Theoret. Comput. Sci.*, 44(1):51–68, 1986. 2

[LLD+04] S. Lengrand, P. Lescanne, D. Dougherty, M. Dezani-Ciancaglini, and S. van Bakel. Intersection types for explicit substitutions. *Inform. and Comput.*, 189(1):17–42, 2004. 4

[LM08] S. Lengrand and A. Miquel. Classical F_ω, orthogonality and symmetric candidates. *Ann. Pure Appl. Logic*, 153:3–20, 2008. 4, 17, 20

[Mel95] P.-A. Melliès. Typed λ-calculi with explicit substitution may not terminate. In M. Dezani-Ciancaglini and G. Plotkin, editors, *Proc. of the 2nd Int. Conf. on Typed Lambda Calculus and Applications (TLCA'95)*, volume 902 of *LNCS*, pages 328–334. Springer-Verlag, 1995. 4, 10

[MM09] G. Munch-Maccagnoni. Focalisation and classical realisability. In E. Grädel and R. Kahle, editors, *Proc. of the 18th Annual Conf. of the European Association for Computer Science Logic (CSL'09)*, volume 5771 of *LNCS*, pages 409–423. Springer-Verlag, 2009. 4, 17

[MV05] P.-A. Melliès and J. Vouillon. Recursive polymorphic types and parametricity in an operational framework. In P. Panangaden, editor, *20th Annual IEEE Symp. on Logic in Computer Science*, pages 82–91. IEEE Computer Society Press, 2005. 4, 17

[NM04] P. M. Neergaard and H. G. Mairson. Types, potency, and idempotency: why nonlinearity and amnesia make a type system work. In C. Okasaki and K. Fisher, editors, *Proc. of the 9th ACM Intern. Conf. on Functional Programming*, pages 138–149. ACM Press, 2004. 2, 31

[Par97] M. Parigot. Proofs of strong normalisation for second order classical natural deduction. *J. of Symbolic Logic*, 62(4):1461–1479, 1997. 4, 17, 20

[Ren11] F. Renaud. *Les ressources explicites vues par la théorie de la réécriture*. PhD thesis, Université Paris 7, 2011. 2, 3, 10, 11

[Sch82] H. Schwichtenberg. Complexity of normalization in the pure typed lambda calculus. In A. S. Troelstra and D. V. Dalen, editors, *The L. E. J. Brouwer Centenary Symposium*. North-Holland, 1982. 3, 32

[Tai75] W. W. Tait. A realizability interpretation of the theory of species. In *Logic Colloquium*, volume 453 of *LNM*, pages 240–251. Springer-Verlag, 1975. 3, 4, 13

[vB92] S. van Bakel. Complete restrictions of the Intersection Type Discipline. *Theoretical Computer Science*, 102(1):135–163, 1992. 6

[vB95] S. van Bakel. Intersection Type Assignment Systems. *Theoret. Comput. Sci.*, 151(2):385–435, 1995. 2

[vRSSX99] F. van Raamsdonk, P. Severi, M. H. B. Sørensen, and H. Xi. Perpetual reductions in λ-calculus. *Inform. and Comput.*, 149(2):173–225, 1999. 29

[Wel02] J. B. Wells. The essence of principal typings. In P. Widmayer, F. T. Ruiz, R. M. Bueno, M. Hennessy, S. Eidenbenz, and R. Conejo, editors, *Proc. of the 29th Intern. Col. on Automata, Languages and Programming (ICALP)*, volume 2380 of *LNCS*, pages 913–925. Springer-Verlag, 2002. 23

A Filter models: classical vs. intuitionistic realisability

The orthogonality method comes from the denotational and operational semantics of symmetric calculi, such as proof-term calculi for classical or linear logic.

In some sense, orthogonality only builds semantics in a continuation passing style, and (as we have seen) this still makes sense for typed λ-calculi that are purely intuitionistic. While this is sufficient to prove important properties of typed λ-terms such as strong normalisation, the models are unable to reflect some of their purely intuitionistic features.

This phenomenon could be seen in presence of a "positive" type (i.e. datatype) $\mathfrak{P}$, for which $[\![\mathfrak{P}]\!]$ is **not** closed under bi-orthogonal and $[\mathfrak{P}]$ is defined as $[\![\mathfrak{P}]\!]^{\perp}$. Model $\mathcal{M}^{\perp\!\!\!\perp}_{\mathcal{F}}$ provides the interpretation

$$\begin{aligned}[\![\mathfrak{P}{\to}\mathfrak{P}]\!] &= ([\![\mathfrak{P}]\!] :: [\mathfrak{P}])^{\perp} = \{u \in \mathcal{D} \mid \forall v \in [\![\mathfrak{P}]\!], \forall \vec{v'} \in [\mathfrak{P}], u \perp\!\!\!\perp v :: \vec{v'}\} \\ &= \{u \in \mathcal{D} \mid \forall v \in [\![\mathfrak{P}]\!], \forall \vec{v'} \in [\mathfrak{P}], u@v \perp\!\!\!\perp \vec{v'}\} \\ &= \{u \in \mathcal{D} \mid \forall v \in [\![\mathfrak{P}]\!], u@v \in [\![\mathfrak{P}]\!]^{\perp\perp}\}\end{aligned}$$

while model $\mathcal{M}^{\mathsf{i}}_{\mathcal{F}}$ would provide

$$[\![\mathfrak{P}{\to}\mathfrak{P}]\!] = \{u \in \mathcal{D} \mid \forall v \in [\![\mathfrak{P}]\!], u@v \in [\![\mathfrak{P}]\!]\}$$

B Preservation of semantics by reduction

When models are built for a typed λ-calculus, it is sometimes expected that the interpretation of terms is preserved under β-reduction (or even β-equivalence). It is not always necessary for the purpose of the model construction (here: proving normalisation properties), and it is clearly not the case for the term models $\mathcal{M}^{\perp\!\!\!\perp}_{\mathsf{SN}}$and $\mathcal{M}^{\perp\!\!\!\perp}_{\cap}$, where terms are interpreted as themselves (at least in the case of the pure λ-calculus). The case of the filter models $\mathcal{M}^{\perp\!\!\!\perp}_{\mathcal{F}}$ and $\mathcal{M}^{\mathsf{i}}_{\mathcal{F}}$ (which heavily rely on Theorem 21) is less obvious. Still we can prove the following:

Theorem 66

1. If $M \longrightarrow_{\beta} M'$, then for all ρ, $[\![M]\!]_{\rho} \subseteq [\![M']\!]_{\rho}$.
2. If $M \rightsquigarrow_{E} M'$, then for all ρ, $[\![M]\!]_{\rho} = [\![M']\!]_{\rho}$.
3. If $M \longrightarrow_{B,S} M'$, then for all ρ, $[\![M]\!]_{\rho} = [\![M']\!]_{\rho}$.

Proof:

1. Corollary of Subject Reduction (Theorem 9).
2. One inclusion is the previous case and the other is a corollary of Subject Expansion (Theorem 51).
3. Same argument, with Subject reduction and Subject Expansion of λS (Theorems 12 and 43) □

Example 1 There are cases where $[\![M]\!]_{\rho} \neq [\![M']\!]_{\rho}$:
an obvious example is when $M \notin \mathsf{SN}^{\lambda}$ but $M' \in \mathsf{SN}^{\lambda}$ (e.g. $M := (\lambda y.z)((\lambda x.xx)(\lambda x.xx))$ and $M' := z$); there exists ρ such that $[\![M]\!]_{\rho} = \bot$ and $[\![M']\!]_{\rho} \neq \bot$.

We can also find an example where $M \longrightarrow_{\beta} M'$, $[\![M]\!]_{\rho} \neq \bot$, and $[\![M]\!]_{\rho} \neq [\![M']\!]_{\rho}$:

Example 2 Take $M := (\lambda z.(\lambda y.a)(zz))$ and $M' := \lambda z.a$.

Proof: Suppose that $[\![M]\!]_{\rho} = [\![M']\!]_{\rho}$ with $\rho = (a \mapsto \top)$. Then:

- $N = \lambda x.xx \in \mathsf{SN}^{\lambda}$ and closed, so $[\![N]\!]_{\rho} \neq \bot$.
- $[\![M]\!]_{\rho}@[\![N]\!]_{\rho} = [\![(\lambda y.a)(zz)]\!]_{\rho,z\mapsto[\![N]\!]_{\rho}} = [\![(\lambda y.a)]\!]_{\rho,z\mapsto[\![N]\!]_{\rho}}@[\![zz]\!]_{\rho,z\mapsto[\![N]\!]_{\rho}}$
  $= [\![\lambda y.a]\!]_{\rho}@[\![NN]\!]_{\rho} = [\![\lambda y.a]\!]_{\rho}@\bot = \bot$
- $[\![M']\!]_{\rho}@[\![N]\!]_{\rho} = [\![a]\!]_{\rho,z\mapsto[\![N]\!]_{\rho}} = \top$

Hence $\top = \bot$, which is a contradiction. □

Notice that this proof only uses the properties expected from the model (Theorem 21 and the characterisation of SN^λ) and not the construction of the model itself.

C Full proofs

Lemma 2 (Properties of $\approx$) For all U, V, W, F, U', V',

1. $\approx$ is an equivalence relation.
2. If $U \approx \omega$, then $U = \omega$ and if $U \approx F$, then $U = F$.
3. $U \cap V \approx V \cap U$ and $(U \cap V) \cap W \approx U \cap (V \cap W)$.
4. If $U \approx U'$ and $V \approx V'$, then $U \cap V \approx U' \cap V'$.
5. For all U and V, if $U \cap V \approx U$, then $V = \omega$.

Proof:

The first four points are proved by straightforward inductions on derivations. The last point is more subtle. We define $\phi(U)$ by induction on U as follows:

$$\begin{array}{lcl} \phi(\omega) & := & 0 \\ \phi(F) & := & 1 \\ \phi(A \cap B) & := & \phi(A) + \phi(B) \end{array}$$

So for all U and V we have $\phi(U \cap V) = \phi(U) + \phi(V)$. Also, for all A, $\phi(A) > 0$. So for all U, if $\phi(U) = 0$, then $U = \omega$.

Moreover, for all U and V, if $U \approx V$, then $\phi(U) = \phi(V)$ (by induction on $U \approx V$).

Now if $U \cap V \approx U$, then

$$\phi(U \cap V) = \phi(U) + \phi(V) = \phi(U)$$

So we have $\phi(V) = 0$, from which we get $V = \omega$.

□

Lemma 6 (Typing of explicit substitution) Assume $\Gamma, x : A \vdash^n_{\cap\subseteq} M : B$ and $\Delta \vdash^m_{\cap\subseteq} N : A$. Then, there exists Γ' such that $\Gamma' \approx \Gamma \cap \Delta$ and $\Gamma' \vdash^{n+m}_{\cap\subseteq} M[x := N] : B$.

Proof:

By induction on B:

- If $B = F$, then the result is trivial : we use the (Subst) rule.
- If $B = B_1 \cap B_2$, then, by Lemma 5.1, there exist Γ_1, Γ_2, A_1, A_2, n_1 and n_2 such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $A = A_1 \cap A_2$, $n = n_1 + n_2$, $\Gamma_1, x : A_1 \vdash^{n_1}_{\cap\subseteq} M : B_1$ and $\Gamma_2, x : A_2 \vdash^{n_2}_{\cap\subseteq} M : B_2$. By hypothesis, $\Delta \vdash^m_{\cap\subseteq} N : A$. Hence, by Lemma 5.1, there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1}_{\cap\subseteq} N : A_1$ and $\Delta_2 \vdash^{m_2}_{\cap\subseteq} N : A_2$.

  By induction hypothesis, there exist Γ'_1 and Γ'_2 such that $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$, $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$, $\Gamma'_1 \vdash^{n_1+m_1}_{\cap\subseteq} M[x := N] : B_1$ and $\Gamma'_2 \vdash^{n_2+m_2}_{\cap\subseteq} M[x := N] : B_2$. So we have $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2}_{\cap\subseteq} M[x := N] : B_1 \cap B_2$ with $n_1+m_1+n_2+m_2 = n+m$, $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) \approx \Gamma \cap \Delta$ and $B = B_1 \cap B_2$.

□

Lemma 8 (Typing of implicit substitutions) If $\Gamma, x : U \vdash^n_{\cap\subseteq} M : A$ and $\Delta \vdash^m_{\cap\subseteq} N : U$, then there exists Γ' such that $\Gamma' \approx \Gamma \cap \Delta$ and $\Gamma' \vdash^{n+m}_{\cap\subseteq} M\{x := N\} : A$.

Proof:

By induction on the derivation of $\Gamma, x : U \vdash_{\cap\subseteq} M : A$.

- $\dfrac{}{x : F \vdash_{\cap\subseteq} x : F}$
  Here $\Gamma = ()$, $x = M$, $n = 0$, $A = F$ and $U = F$. Therefore, $M\{x := N\} = N$ and $\Gamma \cap \Delta = \Delta$. By hypothesis, $\Delta \vdash^{m}_{\cap\subseteq} N : U$. So we have $\Gamma \cap \Delta \vdash^{n+m}_{\cap\subseteq} M\{x := N\} : A$ with $\Gamma \cap \Delta \approx \Gamma \cap \Delta$.

- $\dfrac{}{y : F \vdash_{\cap\subseteq} y : F}$ with $y \neq x$
  Here $\Gamma = (y : F)$, $A = F$, $U = \omega$, $M = y$, $n = 0$. Since $\omega = U$ and $\Delta \vdash^{m}_{\cap\subseteq} N : U$, we have $\Delta = ()$ and $m = 0$. Then, we have, $\Gamma \cap \Delta \vdash^{n+m}_{\cap\subseteq} M\{x := N\} : A$ because $M\{x := N\} = y$.

- $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1}_{\cap\subseteq} M : A_1 \quad \Gamma_2, x : U_2 \vdash^{n_2}_{\cap\subseteq} M : A_2}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2}_{\cap\subseteq} M : A_1 \cap A_2}$
  Here $A = A_1 \cap A_2$, $n = n_1 + n_2$, $U = U_1 \cap U_2$ and $\Gamma = \Gamma_1 \cap \Gamma_2$. By hypothesis, $\Delta \vdash^{m}_{\cap\subseteq} N : U$. So, by Lemma 5.1, there exist Δ_1, Δ_2, m_1, m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $\Delta_1 \vdash^{m_1}_{\cap\subseteq} N : U_1$ and $\Delta_2 \vdash^{m_2}_{\cap\subseteq} N : U_2$. By induction hypothesis, there exists Γ'_1 such that $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$ and $\Gamma'_1 \vdash^{n_1+m_1}_{\cap\subseteq} M\{x := N\} : A_1$. We also have the existence of Γ'_2 such that $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$ and $\Gamma'_2 \vdash^{n_2+m_2}_{\cap\subseteq} M\{x := N\} : A_2$.

  Therefore we have $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2}_{\cap\subseteq} M : A$ with $n_1 + m_1 + n_2 + m_2 = n + m$ and $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) \approx \Gamma \cap \Delta$.

- $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1}_{\cap\subseteq} M_1 : A_1 \to A \quad \Gamma_2, x : U_2 \vdash^{n_2}_{\cap\subseteq} M_2 : A_1}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2+1}_{\cap\subseteq} M : A}$
  Here $M = M_1 M_2$, $n = n_1 + n_2 + 1$, $U = U_1 \cap U_2$ and $\Gamma = \Gamma_1 \cap \Gamma_2$. By hypothesis, $\Delta \vdash^{m}_{\cap\subseteq} N : U$. So, by Lemma 5, there exist Δ_1, Δ_2, m_1, m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1}_{\cap\subseteq} N : U_1$ and $\Delta_2 \vdash^{m_2}_{\cap\subseteq} N : U_2$. By induction hypothesis, there exists Γ'_1 such that $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$ and $\Gamma'_1 \vdash^{n_1+m_1}_{\cap\subseteq} M_1\{x := N\} : A_1 \to A$. We also have the existence of Γ'_2 such that $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$ and $\Gamma'_2 \vdash^{n_2+m_2}_{\cap\subseteq} M_2\{x := N\} : A_1$.

  Therefore we have $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2+1}_{\cap\subseteq} M_1 M_2\{x := N\} : A$ with $n_1 + m_1 + n_2 + m_2 + 1 = n + m$ and $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) \approx \Gamma \cap \Delta$.

- $\dfrac{\Gamma, x : U, y : V \vdash^{n}_{\cap\subseteq} M_1 : F \quad A_1 \subseteq V}{\Gamma, x : U \vdash^{n}_{\cap\subseteq} \lambda y.M_1 : A_1 \to F}$ with $x \neq y$ and $y \notin fv(N)$.
  By induction hypothesis, there exist Γ' such that $(\Gamma, y : V) \cap \Delta \approx \Gamma'$ and $\Gamma' \vdash^{n+m}_{\cap\subseteq} M_1\{x := N\} : F$. $y \notin fv(N)$, so there exist Γ'' and V' such that $\Gamma' \approx (\Gamma'', y : V')$, $V \approx V'$, and $\Gamma'' \approx \Gamma \cap \Delta$. Then we can conclude.

- We do not have to deal with the other rules because they cannot be used for a pure λ-term.

□

Theorem 9 (Subject Reduction for λ) If $\Gamma \vdash^{n}_{\cap\subseteq} M : A$ and $M \longrightarrow_{\beta} M'$, then there exist m and Δ such that $m < n$, $\Gamma \subseteq \Delta$ and $\Delta \vdash^{m}_{\cap\subseteq} M' : A$.

Proof:

First by induction on $M \longrightarrow_{\beta} M'$, then by induction on A.

- If there exist A_1 and A_2 such that $A = A_1 \cap A_2$, then, by Lemma 5.1, there exist Γ_1, Γ_2, n_1, n_2 such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $\Gamma_1 \vdash^{n_1}_{\cap\subseteq} M : A_1$ and $\Gamma_2 \vdash^{n_2}_{\cap\subseteq} M : A_2$. By induction hypothesis (on $(M \longrightarrow_{\beta} M', A_1)$ and $(M \longrightarrow_{\beta} M', A_2)$), there exist Δ_1 and m_1 such that $\Gamma_1 \subseteq \Delta_1$, $m_1 < n_1$ and $\Delta_1 \vdash^{m_1}_{\cap\subseteq} M' : A_1$. We also have the existence of Δ_2 and m_2 such that $m_2 < n_2$, $\Gamma_2 \subseteq \Delta_2$ and $\Delta_2 \vdash^{m_2}_{\cap\subseteq} M' : A_2$. Then we have $\Delta_1 \cap \Delta_2 \vdash^{m_1+m_2}_{\cap\subseteq} M' : A$ with $m_1 + m_2 < n$ and $\Gamma \subseteq \Delta_1 \cap \Delta_2$.

- $\dfrac{}{(\lambda x.M_1)M_2 \longrightarrow_{\beta} M_1\{x := M_2\}}$ and $A = F$
  Then there exist Γ_1, Γ_2, n_1, n_2 and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1}_{\cap\subseteq} \lambda x.M_1 :$

$B \to F$ and $\Gamma_2 \vdash^{n_2}_{\cap\subseteq} M_2 : B$. So there exists U such that $B \subseteq U$ and $\Gamma_1, x : U \vdash^{n_1}_{\cap\subseteq} M_1 : F$. Then, by Lemma 5.4, there exist Γ_2' and n_2' such that $\Gamma_2 \subseteq \Gamma_2'$, $n_2' \leq n_2$ and $\Gamma_2' \vdash^{n_2'}_{\cap\subseteq} M_2 : U$. Therefore, by the substitution lemma (Lemma 8), there exists Γ' such that $\Gamma' \approx \Gamma_1 \cap \Gamma_2'$ and $\Gamma' \vdash^{n_1+n_2'}_{\cap\subseteq} M_1\{x := M_2\} : F$ with $A = F$, $n_1 + n_2' < n$ and $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma_2' \approx \Gamma'$.

- $\dfrac{M_1 \longrightarrow_\beta M_1'}{\lambda x.M_1 \longrightarrow_\beta \lambda x.M_1'}$ and $A = F$
  Here, there exist B, G and U such that $A = B \to G$, $B \subseteq U$ and $\Gamma, x : U \vdash^{n}_{\cap\subseteq} M_1 : G$. By induction hypothesis, there exist Γ', U' and m such that $\Gamma \subseteq \Gamma'$, $U \subseteq U'$, $m < n$ and $\Gamma', x : U' \vdash^{m}_{\cap\subseteq} M_1' : G$. So we have $B \subseteq U'$. Hence $\Gamma' \vdash^{m}_{\cap\subseteq} M' : A$.
- $\dfrac{M_1 \longrightarrow_\beta M_1'}{M_1M_2 \longrightarrow_\beta M_1'M_2}$ and $A = F$
  Then there exist B, Γ_1, Γ_2, n_1, n_2 such that $n = n_1 + n_2 + 1$, $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash^{n_1}_{\cap\subseteq} M_1 : B \to A$ and $\Gamma_2 \vdash^{n_2}_{\cap\subseteq} M_2 : B$. By induction hypothesis there exist Γ_1' and n_1' such that $\Gamma_1 \subseteq \Gamma_1'$, $n_1' < n_1$ and $\Gamma_1' \vdash^{n_1'}_{\cap\subseteq} M_1' : B \to F$. Therefore $\Gamma_1' \cap \Gamma_2 \vdash^{n_1'+n_2+1}_{\cap\subseteq} M_1'M_2 : A$ with $n_1' + n_2 + 1 < n$ and $\Gamma \subseteq \Gamma_1' \cap \Gamma_2$.
- $\dfrac{M_2 \longrightarrow_\beta M_2'}{M_1M_2 \longrightarrow_\beta M_1M_2'}$ and $A = F$
  Then there exist B, Γ_1, Γ_2, n_1, n_2 such that $n = n_1 + n_2 + 1$, $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash^{n_1}_{\cap\subseteq} M_1 : B \to A$ and $\Gamma_2 \vdash^{n_2}_{\cap\subseteq} M_2 : B$. By induction hypothesis there exist Γ_2' and n_2' such that $\Gamma_2 \subseteq \Gamma_2'$, $n_2' < n_2$ and $\Gamma_2' \vdash^{n_2'}_{\cap\subseteq} M_2' : B$. Therefore, $\Gamma_1 \cap \Gamma_2' \vdash^{n_1+n_2'+1}_{\cap\subseteq} M_1M_2' : A$ with $n_1 + n_2' + 1 < n$ and $\Gamma \subseteq \Gamma_1 \cap \Gamma_2'$.

□

Lemma 11 $\longrightarrow_{S,W}$ terminates.

Proof:

By a polynomial argument.

We define $m_x(M)$ as follow: if $x \notin fv(M)$, then $m_x(M) = 1$. Otherwise we have:

$$\begin{array}{lcll}
m_x(x) & = & 1 & \\
m_x(\lambda y.M) & = & m_x(M) & \\
m_x(M_1M_2) & = & m_x(M_1) + m_x(M_2) & x \in fv(M_1), x \in fv(M_2) \\
m_x(M_1M_2) & = & m_x(M_1) & x \notin fv(M_2) \\
m_x(M_1M_2) & = & m_x(M_2) & x \notin fv(M_1) \\
m_x(M[y := N]) & = & m_x(M) + m_y(M) \times (m_x(N) + 1) & x \in fv(M), x \in fv(N) \\
m_x(M[y := N]) & = & m_y(M) \times (m_x(N) + 1) & x \notin fv(M), x \in fv(N) \\
m_x(M[y := N]) & = & m_x(M) & x \notin fv(N)
\end{array}$$

We also define $S(M)$ as follow:

$$\begin{array}{lcl}
S(x) & = & 1 \\
S(M_1M_2) & = & S(M_1) + S(M_2) \\
S(\lambda x.M) & = & S(M) \\
S(M[x := N]) & = & S(M) + m_x(M) \times S(N)
\end{array}$$

Finally, we define $I(M)$ as follow:

$$\begin{array}{lcl}
I(x) & = & 2 \\
I(\lambda x.M) & = & 2I(M) + 2 \\
I(M_1M_2) & = & 2I(M_1) + 2I(M_2) + 2 \\
I(M[x := N]) & = & I(M) \times (I(N) + 1)
\end{array}$$

If we consider $n = (S(M), I(M))$ in lexical order, then $\longrightarrow_{S,W}$ strictly decreases n and $\equiv$ does not change it.

Hence $\longrightarrow_{S,W}$ terminates. This lemma and proof are a special case of [KR11]. □

Theorem 12 (Subject Reduction for λS)
Assume $\Gamma \vdash^{n}_{\cap\subseteq} M : A$. We have the following properties:

- If $M \longrightarrow_B M'$, then there exist Γ' and m such that $\Gamma \subseteq \Gamma'$, $m < n$ and $\Gamma' \vdash^{m}_{\cap\subseteq} M' : A$
- If $M \longrightarrow_S M'$, then there exists Γ' such that $\Gamma \approx \Gamma'$ and $\Gamma' \vdash^{n}_{\cap\subseteq} M' : A$
- If $M \longrightarrow_W M'$, then there exist Γ' and m such that $\Gamma \subseteq \Gamma'$, $m \leq n$ and $\Gamma' \vdash^{m}_{\cap\subseteq} M' : A$
- If $M \equiv M'$, then there exists Γ' such that $\Gamma \approx \Gamma'$ and $\Gamma' \vdash^{n}_{\cap\subseteq} M' : A$

Proof:
First by induction on $M \longrightarrow_E M'$ and $M \equiv M'$, then by induction on A.
For modularity, the triplet $(\longrightarrow_E, R, r)$ can be one of the following triplets: $(\longrightarrow_B, \approx, <)$, $(\longrightarrow_S, \approx, =)$, $(\longrightarrow_W, \subseteq, \leq)$, $(\equiv, \approx, =)$.

- If $A = A_1 \cap A_2$, then there exist Γ_1, Γ_2, n_1 and n_2 such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $\Gamma_1 \vdash^{n_1}_{\cap\subseteq} M : A_1$ and $\Gamma_2 \vdash^{n_2}_{\cap\subseteq} M : A_2$.

  By induction hypothesis (on $(M \longrightarrow_E M', A_1)$ and $(M \longrightarrow_E M', A_2)$), there exist Γ'_1, Γ'_2, m_1 and m_2 such that $\Gamma_1 \; R \; \Gamma'_1$, $\Gamma_2 \; R \; \Gamma'_2$, $m_1 \; r \; n_1$, $m_2 \; r \; n_2$, $\Gamma'_1 \vdash^{m_1}_{\cap\subseteq} M' : A_1$ and $\Gamma'_2 \vdash^{m_2}_{\cap\subseteq} M' : A_2$.

  Hence, $\Gamma'_1 \cap \Gamma'_2 \vdash^{m_1+m_2}_{\cap\subseteq} M' : A_1 \cap A_2$ with $A = A_1 \cap A_2$, $m_1 + m_2 \; r \; n$, $\Gamma \; R \; \Gamma'_1 \cap \Gamma'_2$.
- $(\lambda x.M_1)M_2 \longrightarrow_B M_1[x := M_2]$ and $A = F$: There exist Γ_1, Γ_2, n_1, n_2 and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1}_{\cap\subseteq} \lambda x.M_1 : B \to F$ and $\Gamma_2 \vdash^{n_2}_{\cap\subseteq} M_2 : B$. Hence, there exists U such that $B \subseteq U$ and $\Gamma_1, x : U \vdash^{n_1}_{\cap\subseteq} M_1 : F$.
  - If $U = C$, then by using Lemma 5.4 there exist Δ and m such that $m \leq n_2$, $\Gamma_2 \subseteq \Delta$ and $\Delta \vdash^{m}_{\cap\subseteq} M_2 : C$. Hence $\Gamma_1 \cap \Delta \vdash^{n_1+m}_{\cap\subseteq} M_1[x := M_2] : F$ with $n_1 + m < n$, $F = A$ and $\Gamma \subseteq \Gamma_1 \cap \Delta$.
  - If $U = \omega$, then $\Gamma_1 \cap \Gamma_2 \vdash^{n_1+n_2}_{\cap\subseteq} M_1[x := M_2] : F$ with $n_1 + n_2 < n$, $A = F$ and $\Gamma \subseteq \Gamma_1 \cap \Gamma_2$.
- $y[x := N] \longrightarrow_W y$, $x \neq y$ and $A = F$: there exist Γ_1, Γ_2, n_1, n_2 and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $\Gamma_1 \vdash^{n_1}_{\cap\subseteq} N : B$ and $\Gamma_2, x : \omega \vdash^{n_2}_{\cap\subseteq} y : F$. So we have $\Gamma_2 \vdash^{n_2}_{\cap\subseteq} y : A$ with $\Gamma \subseteq (\Gamma_2, x : \omega)$ and $n_2 \leq n$.
- $x[x := N] \longrightarrow_S N$ and $A = F$: there exist Γ_1, Γ_2, n_1, n_2 and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $\Gamma_1 \vdash^{n_1}_{\cap\subseteq} N : B$ and $\Gamma_2, x : B \vdash^{n_2}_{\cap\subseteq} x : F$. Hence $\Gamma_2 = ()$ and $B = F$ and $n_2 = 0$. Therefore $\Gamma = \Gamma_1$ and $n_1 = n$. So we have $\Gamma \vdash^{n}_{\cap\subseteq} N : A$ with $\Gamma \approx \Gamma$.
- $(M_1M_2)[x := N] \longrightarrow_S M_1[x := N]M_2[x := N]$ with $x \in fv(M_1)$, $x \in fv(M_2)$ and $A = F$: Then there exist Γ_1, Γ_2, Γ_3, Γ_4, A_1, A_2, B, n_1, n_2, n_3 and n_4 such that: $\Gamma_1, x : A_1 \vdash^{n_1}_{\cap\subseteq} M_1 : B \to F$ $\Gamma_2, x : A_2 \vdash^{n_2}_{\cap\subseteq} M_2 : B$, $\Gamma_3 \vdash^{n_3}_{\cap\subseteq} N : A_1$, $\Gamma_4 \vdash^{n_4}_{\cap\subseteq} N : A_2$, $n = n_1 + n_2 + n_3 + n_4$ and $\Gamma = (\Gamma_1 \cap \Gamma_2) \cap (\Gamma_3 \cap \Gamma_4)$. Hence $(\Gamma_1 \cap \Gamma_3) \cap (\Gamma_2 \cap \Gamma_4) \vdash^{n_1+n_3+n_2+n_4}_{\cap\subseteq} M_1[x := N]M_2[x := N] : F$.
- $(M_1M_2)[x := N] \longrightarrow_S M_1[x := N]M_2$ with $x \notin fv(M_2)$ and $A = F$: Then there exist Γ_1, Γ_2, Γ_3, U, A_1, B, n_1, n_2, and n_3 such that: $\Gamma_1, x : U \vdash^{n_1}_{\cap\subseteq} M_1 : B \to F$ $\Gamma_2, x : \omega \vdash^{n_2}_{\cap\subseteq} M_2 : B$, $\Gamma_3 \vdash^{n_3}_{\cap\subseteq} N : A_1$, $n = n_1 + n_2 + n_3$, $\Gamma = (\Gamma_1 \cap \Gamma_2) \cap \Gamma_3$, $U = A_1$ or $U = \omega$. Hence $(\Gamma_1 \cap \Gamma_3) \cap \Gamma_2 \vdash^{n_1+n_3+n_2}_{\cap\subseteq} M_1[x := N]M_2 : F$.
- $(M_1M_2)[x := N] \longrightarrow_S M_1M_2[x := N]$ with $x \notin fv(M_1)$, $x \in fv(M_2)$ and $A = F$: Then there exist Γ_1, Γ_2, Γ_3, A_1, B, n_1, n_2 and n_3 such that $\Gamma_1, x : \omega \vdash^{n_1}_{\cap\subseteq} M_1 : B \to F$, $\Gamma_2, x : A_1 \vdash^{n_2}_{\cap\subseteq} M_2 : F$, $\Gamma_3 \vdash^{n_3}_{\cap\subseteq} N : A_1$, $n = n_1 + n_2 + n_3$, $\Gamma = (\Gamma_1 \cap \Gamma_2) \cap \Gamma_3$, Hence $\Gamma_1 \cap (\Gamma_2 \cap \Gamma_3) \vdash^{n_1+n_2+n_3}_{\cap\subseteq} M_1M_2[x := N] : F$.
- For $M[x := N_1][y := N_2] \equiv M[y := N_2][x := N_1]$ with $x \neq y$, $x \notin fv(N_2)$, $y \notin fv(N_1)$ and $A = F$: There exist Γ_1, Γ_2, n_1, n_2, U and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $U = B$ or $U = \omega$, $\Gamma_1, y : U \vdash^{n_1}_{\cap\subseteq} M[x := N_1] : F$ and $\Gamma_2 \vdash^{n_2}_{\cap\subseteq} N_2 : B$. Therefore, there exist Γ_3, Γ_4, n_3, n_4, V and C such that $\Gamma_1, y : U = \Gamma_3 \cap \Gamma_4$, $n_1 = n_3 + n_4$, $V = C$ or $V = \omega$, $\Gamma_3, x : V \vdash^{n_3}_{\cap\subseteq} M : F$ and $\Gamma_4 \vdash^{n_4}_{\cap\subseteq} N_1 : C$. By the fact that $y \notin fv(N_1)$ and by Lemma 5.2, we have $\Gamma_4 = \Gamma_4, x : \omega$. Hence, there exists Γ_5 such that $\Gamma_3 = \Gamma_5, x : U$ and $\Gamma_1 = \Gamma_5 \cap \Gamma_4$. So,

$\Gamma_5, y:U, x:V \vdash^{n_3}_{\cap\subseteq} M:F$. Hence, $(\Gamma_5, x:V) \cap \Gamma_2 \vdash^{n_3+n_2}_{\cap\subseteq} M[y := N_2]:F$. By the fact that $x \notin fv(N_2)$ and by Lemma 5, $\Gamma_2 = \Gamma_2, x:\omega$. Hence, $(\Gamma_5, x:V) \cap \Gamma_2 = \Gamma_5 \cap \Gamma_2, x:V$. Therefore, $(\Gamma_5 \cap \Gamma_2)\Gamma_4 \vdash^{n_3+n_2+n_4}_{\cap\subseteq} M[y := N_2][x := N_1]:F$ with $n_3 + n_2 + n_4 = n$ and $(\Gamma_5 \cap \Gamma_2) \cap \Gamma_4 \approx \Gamma$.

- The other rules follow the same patterns, especially for the propagation of an explicit substitution over another explicit substitution. Now concerning the congruent closure of the rules, all cases are straightforward but for the following one:
- $M[x := N] \longrightarrow_W M'[x := N]$ with $M \longrightarrow_W M'$, $x \in fv(M)$, $x \in fv(M')$ and $A = F$: Then there exist Γ_1, Γ_2, B, n_1, n_2 such that: $\Gamma_1, x:B \vdash^{n_1}_{\cap\subseteq} M:F$ and $\Gamma_2 \vdash^{n_2}_{\cap\subseteq} N:B$, $n = n_1 + n_2$ and $\Gamma = \Gamma_1 \cap \Gamma_2$. By induction hypothesis, there exist Γ'_1, C and n'_1 such that $\Gamma_1 \subseteq \Gamma'_1$, $B \subseteq C$, $n'_1 \leq n_1$ and $\Gamma'_1, x:C \vdash^{n'_1}_{\cap\subseteq} M':A$. Then there exist Γ'_2 and n'_2 such that $\Gamma_2 \subseteq \Gamma'_2$, $n'_2 \leq n_2$ and $\Gamma'_2 \vdash^{n'_2}_{\cap\subseteq} N:C$. Hence $\Gamma'_1 \cap \Gamma'_2 \vdash^{n'_1+n'_2}_{\cap\subseteq} M'[x := N]:F$ with $n'_1 + n'_2 \leq n$ and $\Gamma \subseteq \Gamma'_1 \cap \Gamma'_2$.

□

Theorem 5 (Subject Reduction for λlxr) If $\Gamma \vdash^n_{\cap\subseteq} M:A$ then:

- If $M \longrightarrow_B M'$, then there exist Γ' and m such that $\Gamma \subseteq \Gamma'$, $m < n$, and $\Gamma' \vdash^m_{\cap\subseteq} M':A$
- If $M \longrightarrow_E M'$ and $B \notin E$, then there exist Γ' and m such that $\Gamma \subseteq \Gamma'$, $m \leq n$ and $\Gamma' \vdash^m_{\cap\subseteq} M':A$.
- If $M \equiv M'$ then there exist Γ' such that $\Gamma \approx \Gamma'$ and $\Gamma' \vdash^n_{\cap\subseteq} M':A$.

Proof: First by induction on $M \longrightarrow_B M'$ (resp. $M \longrightarrow_E M'$, $M \equiv M'$) then by induction on A. The proof is similar to the proof of Subject reduction. Here we will only detail the cases of rules D, W, B and $Merge$:

- For $C^{y,z}_x(M)[x := N] \longrightarrow C^{Y,Z}_X(M[y := N_1][z := N_2])$ with $A = F$: There exist n_1, n_2, n_3, B_1, C_1, Δ, $\vec{B}$ and $\vec{C}$ such that $\Gamma = \Delta, \vec{X}:\vec{B} \cap \vec{C}$, $n = n_1 + n_2 + n_3$, $\Delta, y:B_1, z:C_1 \vdash^{n_1}_{\cap\subseteq} M:F$, $\vec{X}:\vec{B} \vdash^{n_2}_{\cap\subseteq} N:B_1$ and $\vec{X}:\vec{C} \vdash^{n_3}_{\cap\subseteq} N:C_1$. Hence, $\vec{Y}:\vec{B} \vdash^{n_2}_{\cap\subseteq} N_1:B_1$ and $\vec{Z}:\vec{C} \vdash^{n_3}_{\cap\subseteq} N_2:C_1$. Therefore, $\Delta, \vec{Y}:\vec{B}, \vec{Z}:\vec{C} \vdash^n_{\cap\subseteq} M[y := N_1][z := N_2]:F$. Then we can conclude.
- For $W_x(M)[x := N] \longrightarrow W_{fv(N)}(M)$ with $X = fv(N)$ and $A = F$: Then, there exist n_1, n_2, Δ, $\vec{B}$ and B_1 such that $n = n_1 + n_2$, $\Gamma = \Delta, \vec{X}:\vec{B}$, $\Delta \vdash^{n_1}_{\cap\subseteq} M:F$, $x \notin \mathsf{Dom}(\Delta)$, and $\vec{X}:\vec{B} \vdash^{n_2}_{\cap\subseteq} N:B_1$. Hence, for all $y \in X$, $y \notin \mathsf{Dom}(\Delta)$. Then we can conclude.
- For $(\lambda x.M)N \longrightarrow_B M[x := N]$, the proof is exactly as the proof for the rule B in λS.
- For $C^{y,z}_w(W_y(M)) \longrightarrow R^z_w(M)$ with $A = F$: Then, there exist Γ_1, A and B such that $\Gamma = \Gamma_1, w:A \cap B$ and $\Gamma_1, y:A, z:B \vdash^n_{\cap\subseteq} W_y(M):F$. Hence, $\Gamma_1, z:B \vdash^n_{\cap\subseteq} M:F$. Therefore, $\Gamma_1, w:B \vdash^n_{\cap\subseteq} R^z_w(M):F$ with $\Gamma \subseteq (\Gamma_1, w:B)$.
- For $C^{x,v}_w(C^{y,z}_x(M)) \equiv C^{x,y}_w(C^{z,v}_x(M))$ with $F = A$: Then, there exist Γ_1, A_1, A_2, A_3, such that $\Gamma = \Gamma_1, w:(A_1 \cap A_2) \cap A_3$ and $\Gamma_1, y:A_1, z:A_2, v:A_3 \vdash^n_{\cap\subseteq} M:F$. Therefore $\Gamma_1, w:(A_2 \cap A_3) \cap A_1 \vdash^n_{\cap\subseteq} C^{x,y}_w(C^{z,v}_x(M)):F$.

□

Theorem 21 (Inductive characterisation of the interpretation)

1. $[\![x]\!]_\rho = \rho(x)$
2. $[\![MN]\!]_\rho = [\![M]\!]_\rho @ [\![N]\!]_\rho$
3. $[\![\lambda x.M]\!]_\rho @ u = [\![M]\!]_{\rho, x \mapsto u}$ if $u \neq \bot$.
4. $[\![M[x := N]]\!]_\rho = [\![M]\!]_{\rho, x \mapsto [\![N]\!]_\rho}$ if $[\![N]\!]_\rho \neq \bot$
5. $[\![W_x(M)]\!]_\rho = [\![M]\!]_\rho$ if $\rho(x) \neq \bot$.
6. $[\![C^{y,z}_x(M)]\!]_\rho = [\![M]\!]_{\rho, y \mapsto \rho(x), z \mapsto \rho(x)}$.

Proof: To prove equalities between I-filters, we only have to prove that they have the same F-types.

1. If $F \in [\![x]\!]_\rho$, then there exists $\Gamma \in \rho$ such that $\Gamma \vdash_{\cap\subseteq} x : F$, so $\Gamma = (x : F)$ so $F \in \rho(x)$. Conversely, if $F \in \rho(x)$, then $(x:F) \in \rho$ and $x:F \vdash_{\cap\subseteq} x:F$. So $[\![x]\!]_\rho = \rho(x)$.

2. Let $F \in [\![MN]\!]_\rho$. There exists $\Gamma \in \rho$ such that $\Gamma \vdash_{\cap\subseteq} MN:F$. Hence, there exist Γ_1, Γ_2 and A such that $\Gamma_1 \vdash_{\cap\subseteq} M:A \to F$ and $\Gamma_2 \vdash_{\cap\subseteq} N:A$ and $\Gamma = \Gamma_1 \cap \Gamma_2$. So $\Gamma \subseteq \Gamma_1$, and $\Gamma_1 \in \rho$. Hence, $A \to F \in [\![M]\!]_\rho$. We also have $A \in [\![N]\!]_\rho$. So we have $F \in [\![M]\!]_\rho @ [\![N]\!]_\rho$.

   Conversely, let $F \in [\![M]\!]_\rho @ [\![N]\!]_\rho$. There exists A such that $A \to F \in [\![M]\!]_\rho$ and $A \in [\![N]\!]_\rho$. So there exists $\Gamma \in \rho$ such that $\Gamma \vdash_{\cap\subseteq} M:A \to F$ and there exists $\Delta \in \rho$ such that $\Delta \vdash_{\cap\subseteq} N:A$. Hence, $\Gamma \cap \Delta \in \rho$ and $\Gamma \cap \Delta \vdash_{\cap\subseteq} MN:F$. So $F \in [\![MN]\!]_\rho$.

3. Let $F \in [\![\lambda x.M]\!]_\rho @ u$. There exists $A \in u$ such that $A \to F \in [\![\lambda x.M]\!]_\rho$. So there exists $\Gamma \in \rho$ such that $\Gamma \vdash_{\cap\subseteq} \lambda x.M : A \to F$. Hence, there exists U such that $A \subseteq U$ and $\Gamma, x:U \vdash_{\cap\subseteq} M:F$.
   - If $U = \omega$, then $(\Gamma, x:U) = \Gamma \in \rho$. Since $\rho \subseteq (\rho, x \mapsto u)$, we have $(\Gamma, x:U) \in (\rho, x \mapsto u)$.
   - If not, then $U \in u$, and we have $(\Gamma, x:U) \in (\rho, x \mapsto u)$.

   So we have $F \in [\![M]\!]_{\rho, x\mapsto u}$.

   Conversely, let $F \in [\![M]\!]_{\rho,x\mapsto u}$. There exists $\Gamma \in (\rho, x \mapsto u)$ such that $\Gamma \vdash_{\cap\subseteq} M:F$.
   - If $x \in fv(M)$, then there exist $\Gamma_1 \in \rho$ and $A \in u$ such that $\Gamma = \Gamma_1, x : A$ (using Lemma 5.1). So we have $\Gamma_1 \vdash_{\cap\subseteq} \lambda x.M : A \to F$, and then $A \to F \in [\![\lambda x.M]\!]_\rho$. Hence, we have $F \in [\![\lambda x.M]\!]_\rho @ u$.
   - If $x \notin fv(M)$, then for all $y \in \mathsf{Dom}(\Gamma)$, $y \neq x$ (using Lemma 5.1). So $\Gamma \in \rho$ and $\Gamma, x : \omega \vdash_{\cap\subseteq} M : F$. Since u is a value, there exist $A \in u$ and $\Gamma \vdash_{\cap\subseteq} \lambda x.M : A \to F$. Hence, $A \to F \in [\![\lambda x.M]\!]_\rho$ and finally $F \in [\![\lambda x.M]\!]_\rho @ u$.

4. Let $F \in [\![M[x := N]]\!]_\rho$. So there exists $\Gamma \in \rho$ such that $\Gamma \vdash_{\cap\subseteq} M[x := N]:F$. Hence there exist Γ_1, Γ_2, A and U such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_{\cap\subseteq} N:A$, $\Gamma_2, x:U \vdash_{\cap\subseteq} M:F$ and $U = A$ or $U = \omega$. Hence $\Gamma_1 \in \rho$ and then $A \in [\![N]\!]_\rho$. Therefore because $\Gamma_2 \in \rho$ we also have $(\Gamma_2, x:A) \in (\rho, x \mapsto [\![N]\!]_\rho)$ and $(\Gamma_2, x:\omega) \in (\rho, x \mapsto [\![N]\!]_\rho)$. Hence $(\Gamma_2, x:U) \in (\rho, x \mapsto [\![N]\!]_\rho)$. So we have $F \in [\![M]\!]_{\rho, x\mapsto [\![N]\!]_\rho}$.

   Conversely, if $F \in [\![M]\!]_{\rho, x\mapsto [\![N]\!]_\rho}$ then there exists $\Gamma \in (\rho, x \mapsto [\![N]\!]_\rho)$ such that $\Gamma \vdash_{\cap\subseteq} M:F$. Hence there exist Γ' and U such that $\Gamma' \in \rho$ and $U \in [\![N]\!]_\rho$ or $U = \omega$.
   - If $U \in [\![N]\!]_\rho$ there exist A and Δ such that $\Delta \vdash_{\cap\subseteq} N:A$ and $A = U$.
   - If $U = \omega$, then because $[\![N]\!]_\rho \neq \bot$ there exists $A \in [\![N]\!]_\rho$. Therefore there exist $\Delta \in \rho$ such that $\Delta \vdash_{\cap\subseteq} N:A$.

   Hence $\Delta \cap \Gamma' \vdash_{\cap\subseteq} M[x := N]:F$ with $(\Delta \cap \Gamma') \in \rho$. Therefore $F \in [\![M[x := N]]\!]_\rho$.

5. Let $F \in [\![W_x(M)]\!]_\rho$. So there exists $\Gamma \in \rho$ such that $\Gamma \vdash_{\cap\subseteq} W_x(M):F$. So there exist Γ', U and A such that $\Gamma = (\Gamma', x:U \cap A)$ and $\Gamma', x:U \vdash_{\cap\subseteq} M:F$. Therefore, $\Gamma \subseteq (\Gamma', x:U)$. Hence $(\Gamma', x:U) \in \rho$. Therefore $F \in [\![M]\!]_\rho$.

   Conversely, if $F \in [\![M]\!]_\rho$, then there exists $\Gamma \in \rho$ such that $\Gamma \vdash_{\cap\subseteq} M:F$. $\rho(x) \neq \bot$, so there exists $A \in \rho(x)$. Also, there exist Γ' and U such that $\Gamma = (\Gamma', x:U)$ and $U = \omega$ or $U \in \rho(x)$. So we have $U \cap A \in \rho(x)$ and $\Gamma' \in \rho$. Hence $(\Gamma, x:U \cap A) \in \rho$ and $\Gamma, x:U \cap A \vdash_{\cap\subseteq} W_x(M):F$. Therefore $F \in [\![W_x(M)]\!]_\rho$.

6. Let $F \in [\![C_x^{y,z}(M)]\!]_\rho$. So there exists $\Gamma \in \rho$ such that $\Gamma \vdash_{\cap\subseteq} C_x^{y,z}(M):F$. Hence there exist Γ', U, V_1 and V_2 such that $\Gamma = (\Gamma', x:U \cap (V_1 \cap V_2))$ and $\Gamma', x:U, y:V_1, z:V_2 \vdash_{\cap\subseteq} M:F$. Therefore $\Gamma' \in \rho$ and $U \cap (V_1 \cap V_2) \in \rho(x)$ or $U \cap (V_1 \cap V_2) = \omega$. So U, V_1 and V_2 are either equal to ω or are in $\rho(x)$. Hence $(\Gamma', x:U, y:V_1, z:V_2) \in (\rho, y \mapsto \rho(x), z \mapsto \rho(x))$. Therefore $F \in [\![M]\!]_{\rho, y\mapsto\rho(x), z\mapsto\rho(x)}$.

   Conversely, if $F \in [\![M]\!]_{\rho, y\mapsto\rho(x), z\mapsto\rho(x)}$, then there exists $\Gamma \in (\rho, x \mapsto \rho(x), y \mapsto \rho(y))$. So there exist Γ', U, V_1 and V_2 such that $\Gamma = (\Gamma', x : U, y : V_1, z : V_2)$ and U, V_1 and V_2 are

either equal to ω or are in $\rho(x)$. Hence $(U \cap (V_1 \cap V_2)) \in \rho(x)$ or is equal to ω. So we have $\Gamma', x{:}U \cap (V_1 \cap V_2) \vdash_{\cap\subseteq} C_x^{y,z}(M){:}F$ with $(\Gamma', x{:}U \cap (V_1 \cap V_2)) \in \rho$. Therefore $F \in [\![C_x^{y,z}(M)]\!]_\rho$. □

Lemma 28 (Adequacy Lemma)
If $\mathfrak{G} \vdash_{\mathcal{S}} M{:}\mathfrak{A}$, then for all valuations σ and for all mappings $\rho \in [\![\mathfrak{G}]\!]_\sigma$ we have $[\![M]\!]_\rho \in [\![\mathfrak{A}]\!]_\sigma$.

Proof: By induction on the derivation of $\mathfrak{G} \vdash_{\mathcal{S}} M : \mathfrak{A}$, using the axioms (A1),...,(A6) from Definition 19. Let σ be a valuation.

- $\dfrac{}{\mathfrak{G}, x{:}\mathfrak{A} \vdash_{\cap\subseteq} x{:}\mathfrak{A}}$
  Let $\rho \in [\![\mathfrak{G}, x{:}\mathfrak{A}]\!]_\sigma$ and let $\overrightarrow{v} \in [\mathfrak{A}]_\sigma$. By definition, $\rho(x) \in [\![\mathfrak{A}]\!]_\sigma$ so $\rho(x) \perp\!\!\!\perp \overrightarrow{v}$, and by axiom (A1) we have $[\![x]\!]_\rho \perp\!\!\!\perp \overrightarrow{v}$. Hence, $[\![x]\!]_\rho \in [\![\mathfrak{A}]\!]_\sigma$.
- $\dfrac{\mathfrak{G}, x{:}\mathfrak{A} \vdash_{\cap\subseteq} M{:}\mathfrak{B}}{\mathfrak{G} \vdash_{\cap\subseteq} \lambda x.M{:}\mathfrak{A}{\to}\mathfrak{B}}$
  Let $\rho \in [\![\mathfrak{G}]\!]_\sigma$ and let $w :: \overrightarrow{v} \in [\mathfrak{A}{\to}\mathfrak{B}]_\sigma = [\![\mathfrak{A}]\!]_\sigma :: [\mathfrak{B}]_\sigma$. As $w \in [\![\mathfrak{A}]\!]_\sigma$, we have $(\rho, x \mapsto w) \in [\![\mathfrak{G}, x{:}\mathfrak{A}]\!]_\sigma$, so by induction hypothesis we have $[\![M]\!]_{\rho, x\mapsto w} \in [\![\mathfrak{B}]\!]_\sigma$. From this we get $[\![M]\!]_{\rho,x\mapsto w} \perp\!\!\!\perp \overrightarrow{v}$ and by axiom (A3) we have $[\![\lambda x.M]\!]_\rho \perp\!\!\!\perp w :: \overrightarrow{v}$. Hence, $[\![\lambda x.M]\!]_\rho \in [\![\mathfrak{A}{\to}\mathfrak{B}]\!]_\sigma$.
- $\dfrac{\mathfrak{G} \vdash_{\cap\subseteq} M{:}\mathfrak{A}{\to}\mathfrak{B} \quad \mathfrak{G} \vdash_{\cap\subseteq} N{:}\mathfrak{A}}{\mathfrak{G} \vdash_{\cap\subseteq} M\ N{:}\mathfrak{B}}$
  Let $\rho \in [\![\mathfrak{G}]\!]_\sigma$ and let $\overrightarrow{v} \in [\mathfrak{B}]_\sigma$. By induction hypothesis we have $[\![N]\!]_\rho \in [\![\mathfrak{A}]\!]_\sigma$ and $[\![M]\!]_\rho \in [\![\mathfrak{A}{\to}\mathfrak{B}]\!]_\sigma$. We thus get $[\![N]\!]_\rho :: v \in [\![\mathfrak{A}]\!]_\sigma :: [\mathfrak{B}]_\sigma = [\mathfrak{A}{\to}\mathfrak{B}]_\sigma$. So $[\![M]\!]_\rho \perp\!\!\!\perp [\![N]\!]_\rho :: v$ and by axiom (A2) we have $[\![M\ N]\!]_\rho \perp\!\!\!\perp v$. Hence $[\![M\ N]\!]_\rho \in [\![\mathfrak{B}]\!]_\sigma$.
- $\dfrac{\mathfrak{G} \vdash_{\cap\subseteq} M{:}\mathfrak{A} \quad \mathfrak{G} \vdash_{\cap\subseteq} M{:}\mathfrak{B}}{\mathfrak{G} \vdash_{\cap\subseteq} M{:}\mathfrak{A} \cap \mathfrak{B}}$
  Let $\rho \in [\![\mathfrak{G}]\!]_\sigma$ and let $\overrightarrow{v} \in [\mathfrak{A} \cap \mathfrak{B}]_\sigma = [\mathfrak{A}]_\sigma \cup [\mathfrak{B}]_\sigma$. By induction hypothesis we have $[\![M]\!]_\rho \in [\![\mathfrak{A}]\!]_\sigma$ and $[\![M]\!]_\rho \in [\![\mathfrak{B}]\!]_\sigma$ so in any case $[\![M]\!]_\rho \perp\!\!\!\perp v$. Hence $[\![M]\!]_\rho \in [\![\mathfrak{A} \cap \mathfrak{B}]\!]_\sigma$.
- $\dfrac{\mathfrak{G} \vdash_{\cap\subseteq} M{:}\mathfrak{A} \cap \mathfrak{B}}{\mathfrak{G} \vdash_{\cap\subseteq} M{:}\mathfrak{A}}$
  Let $\rho \in [\![\mathfrak{G}]\!]_\sigma$ and let $\overrightarrow{v} \in [\mathfrak{A}]_\sigma \subseteq [\mathfrak{A} \cap \mathfrak{B}]_\sigma$. By induction hypothesis we have $[\![M]\!]_\rho \in [\![\mathfrak{A} \cap \mathfrak{B}]\!]_\sigma$ so $[\![M]\!]_\rho \perp\!\!\!\perp v$. Hence $[\![M]\!]_\rho \in [\![\mathfrak{A}]\!]_\sigma$.
- $\dfrac{\mathfrak{G} \vdash_{\cap\subseteq} M{:}\mathfrak{A} \cap \mathfrak{B}}{\mathfrak{G} \vdash_{\cap\subseteq} M{:}\mathfrak{B}}$
  Let $\rho \in [\![\mathfrak{G}]\!]_\sigma$ and let $\overrightarrow{v} \in [\mathfrak{B}]_\sigma \subseteq [\mathfrak{A} \cap \mathfrak{B}]_\sigma$. By induction hypothesis we have $[\![M]\!]_\rho \in [\![\mathfrak{A} \cap \mathfrak{B}]\!]_\sigma$ so $[\![M]\!]_\rho \perp\!\!\!\perp v$. Hence $[\![M]\!]_\rho \in [\![\mathfrak{B}]\!]_\sigma$.
- $\dfrac{\mathfrak{G} \vdash_{\cap\subseteq} M{:}\mathfrak{A}}{\mathfrak{G} \vdash_{\cap\subseteq} M{:}\forall\alpha\mathfrak{A}}\ \alpha \notin ftv(\mathfrak{G})$
  Let $\rho \in [\![\mathfrak{G}]\!]_\sigma$ and let $\overrightarrow{v} \in [\forall\alpha\mathfrak{A}]_\sigma = \bigcup_{Y\subseteq\mathcal{D}^*} [\mathfrak{A}]_{\sigma,\alpha\mapsto Y}$. By induction hypothesis we have $[\![M]\!]_\rho \in [\![\mathfrak{A}]\!]_{\sigma,\alpha\mapsto Y'}$ for all $Y' \subseteq \mathcal{D}^*$, so in any case $[\![M]\!]_\rho \perp\!\!\!\perp v$. Hence $[\![M]\!]_\rho \in [\![\forall\alpha\mathfrak{A}]\!]_\sigma$.
- $\dfrac{\mathfrak{G} \vdash_{\cap\subseteq} M{:}\forall\alpha\mathfrak{A}}{\mathfrak{G} \vdash_{\cap\subseteq} M{:}\mathfrak{A}\{\alpha := \mathfrak{B}\}}$
  Let $\rho \in [\![\mathfrak{G}]\!]_\sigma$ and let $\overrightarrow{v} \in [\mathfrak{A}\{\alpha := \mathfrak{B}\}]_\sigma = [\mathfrak{A}]_{\sigma,\alpha\mapsto[\mathfrak{B}]_\sigma} \subseteq [\forall\alpha\mathfrak{A}]_\sigma$. By induction hypothesis we have $[\![M]\!]_\rho \in [\![\forall\alpha\mathfrak{A}]\!]_\sigma$ so $[\![M]\!]_\rho \perp\!\!\!\perp v$. Hence $[\![M]\!]_\rho \in [\![\mathfrak{A}\{\alpha := \mathfrak{B}\}]\!]_\sigma$.
- $\dfrac{\mathfrak{G} \vdash_{\cap\subseteq} N{:}\mathfrak{A} \quad \mathfrak{G}, x{:}\mathfrak{A} \vdash_{\cap\subseteq} M{:}\mathfrak{B}}{\mathfrak{G} \vdash_{\cap\subseteq} M[x := N]{:}\mathfrak{B}}$
  Let $\rho \in [\![\mathfrak{G}]\!]_\sigma$ and let $\overrightarrow{v} \in [\mathfrak{B}]_\sigma$. By induction hypothesis we have $[\![N]\!]_\rho \in [\![\mathfrak{A}]\!]_\sigma$; therefore

$[\![N]\!]_\rho$ is a value and $(\rho, x \mapsto [\![N]\!]_\rho) \in [\![\mathfrak{G}, x{:}\mathfrak{A}]\!]_\sigma$. By induction hypothesis again we have $[\![M]\!]_{\rho,x\mapsto[\![N]\!]_\rho} \in [\![\mathfrak{B}]\!]_\sigma$. So $[\![M]\!]_{\rho,x\mapsto[\![N]\!]_\rho} \perp\!\!\!\perp \overrightarrow{v}$ and by axiom (A4) we have $[\![M[x := N]]\!]_\rho \perp\!\!\!\perp \overrightarrow{v}$.

- $\dfrac{\mathfrak{G} \vdash_{\cap\subseteq} M{:}\mathfrak{B} \quad x \notin dom(\mathfrak{G})}{\mathfrak{G}, x{:}\mathfrak{A} \vdash_{\cap\subseteq} W_x(M){:}\mathfrak{B}}$

  Let $(\rho, x \mapsto u) \in [\![\mathfrak{G}, x{:}\mathfrak{A}]\!]_\sigma$ and let $\overrightarrow{v} \in [\mathfrak{B}]_\sigma$. We have $\rho \in [\![\mathfrak{G}]\!]_\sigma$ and by induction hypothesis we have $[\![M]\!]_\rho \in [\![\mathfrak{B}]\!]_\sigma$. So $[\![M]\!]_\rho \perp\!\!\!\perp \overrightarrow{v}$ and by axiom (A5) we have $[\![W_x(M)]\!]_{\rho,x\mapsto u} \perp\!\!\!\perp \overrightarrow{v}$.

- $\dfrac{\mathfrak{G}, y{:}\mathfrak{A}, z{:}\mathfrak{A} \vdash_{\cap\subseteq} M{:}\mathfrak{B}}{\mathfrak{G}, x{:}\mathfrak{A} \vdash_{\cap\subseteq} C_x^{y,z}(M){:}\mathfrak{B}}$

  Let $(\rho, x \mapsto u) \in [\![\mathfrak{G}, x{:}\mathfrak{A}]\!]_\sigma$ and let $\overrightarrow{v} \in [\mathfrak{B}]_\sigma$. We have $(\rho, y \mapsto u, z \mapsto u) \in [\![\mathfrak{G}]\!]_\sigma$ and by induction hypothesis we have $[\![M]\!]_{\rho,y\mapsto u,z\mapsto u} \in [\![\mathfrak{B}]\!]_\sigma$. So $[\![M]\!]_{\rho,y\mapsto u,z\mapsto u} \perp\!\!\!\perp \overrightarrow{v}$ and by axiom (A6) we have $[\![W_x(M)]\!]_{\rho,x\mapsto u} \perp\!\!\!\perp \overrightarrow{v}$.

□

Lemma 56 (Most inefficient reduction) *Assume $\Gamma \vdash^n_{\mathsf{opt}} M{:}A$. If M can be reduced by $\longrightarrow_B$ and not by $\longrightarrow_S$, then there exist M' and Γ' such that $\Gamma \approx \Gamma'$, $M \longrightarrow_B M'$ and $\Gamma' \vdash^{n-1}_{\mathsf{opt}} M'{:}A$.*

Proof: We follow the induction given in the proof of Subject Reduction (Theorem 12). In this induction, n can be decreased by more than 1 by a $\longrightarrow_B$ in two cases:

- In the case where the type is an intersection, then n will be decreased by at least 2.
- When we build a typing of $M[x := N]$ from a typing of $(\lambda x.M)N$: if there were subsumption in the typing the λ-abstraction, then the proof calls Lemma 5.4 which might decrease n by more than 1.

Those two cases are never encountered when optimality is assumed, as we prove the result by induction on M. Since M cannot be reduced by $\longrightarrow_S$, it is of one of the following forms:

- $\lambda x.M_1$. It is clear that M_1 satisfies the necessary conditions to apply the induction hypothesis.
- $(\lambda x.M_1)N_1 \dots N_p$ (with $p \geq 1$). We reduce to $M_1[x := N_1]\ N_2 \dots N_p$. By the optimality property, A is not an intersection, and none of the types of $((\lambda x.M_1)N_1 \dots N_i)_{1\leq i\leq p-1}$ are intersections either (since they are applied to an argument). Also by the optimality property, there is no subsumption in the typing of the λ-abstraction, and therefore the call to Lemma 5.4 is replaced by a call to Lemma 5.4 and therefore n is decresed by exactly 1.
- $x[y_1 := N_1]...[y_p := N_p]N_{p+1}...M_m$. Therefore there exists i such that N_i can be reduced by $\longrightarrow_B$. Moreover, optimality requires the type of x to be of the form $A_1^+ \to \cdots \to A_p^+ \to B-$, and therefore the sub-derivation typing N_i is also optimal: we can apply the induction hypothesis on it.

□